2

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

THE USE OF CHAOS METRICS
TO ANALYZE
LAGRAGIAN PARTICLE DIFFUSION MODELS

by

Korey V. Jackson

June, 1992

Thesis Advisor: Ray F. Kamada

[illegible]

Unclassified
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED		1b. RESTRICTIVE MARKINGS			
2a. SECURITY CLASSIFICATION AUTHORITY		3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution is unlimited.			
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE					
4. PERFORMING ORGANIZATION REPORT NUMBER(S)		5. MONITORING ORGANIZATION REPORT NUMBER(S)			
6a. NAME OF PERFORMING ORGANIZATION Naval Postgraduate School	6b. OFFICE SYMBOL *(If applicable)* 33	7a. NAME OF MONITORING ORGANIZATION Naval Postgraduate School			
6c. ADDRESS *(City, State, and ZIP Code)* Monterey, CA 93943-5000		7b. ADDRESS *(City, State, and ZIP Code)* Monterey, CA 93943-5000			
8a. NAME OF FUNDING/SPONSORING ORGANIZATION	8b. OFFICE SYMBOL *(If applicable)*	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER			
8c. ADDRESS *(City, State, and ZIP Code)*		10. SOURCE OF FUNDING NUMBERS			
		Program Element No.	Project No	Task No.	Work Unit Accession Number

11. TITLE *(Include Security Classification)*
THE USE OF CHAOS METRICS TO ANALYZE LAGRANGIAN PARTICLE DIFFUSION MODELS

12. PERSONAL AUTHOR(S) Jackson, Korey V.

13a. TYPE OF REPORT Master's Thesis	13b. TIME COVERED From To	14. DATE OF REPORT *(year, month, day)* June 1992	15. PAGE COUNT 162

16. SUPPLEMENTARY NOTATION
The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

17. COSATI CODES			18. SUBJECT TERMS *(continue on reverse if necessary and identify by block number)*
FIELD	GROUP	SUBGROUP	chaos, particle diffusion, modeling, self-affine fractal dimension, entropy, Lyapunov exponent

19. ABSTRACT *(continue on reverse if necessary and identify by block number)*

Chaos metrics are examined as a tool to analyze atmospheric three-dimensional dispersion models at the individual particle rather than the aggregate level. These include the self-affine fractal dimension, D_A, Shannon entropy, S, and Lyapunov exponent, λ. Intercomparison of these metrics is first performed with the one-dimensional logistics difference equation and the two-dimensional Henon systems of equations. The fractal dimension and Shannon entropy are then measured as a function of the inverse Monin-Obukhov length (1/L) for two three-dimensional Lagrangian particle dispersion models, the McNider particle dispersion model and the NPS particle dispersion model now under development. The fractal dimension and Shannon entropy uncover weaknesses in both models which are not obvious with standard geophysical measures. They also reveal similarities and difference between the atmospheric models and simple chaos systems. Combined, these chaos measures may lend detailed insight into the behavior of Lagrangian Monte Carlo dispersion models in general.

20 DISTRIBUTION/AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS REPORT ☐ DTIC USERS	21 ABSTRACT SECURITY CLASSIFICATION Unclassified	
22a. NAME OF RESPONSIBLE INDIVIDUAL Ray F. Kamada	22b TELEPHONE *(Include Area code)* (408) 646-2674	22c OFFICE SYMBOL PHKD

DD FORM 1473, 84 MAR 83 APR edition may be used until exhausted
All other editions are obsolete

SECURITY CLASSIFICATION OF THIS PAGE
Unclassifed

The Use of Chaos Metrics to Analyze Lagrangian Particle Diffusion Models

by

Korey V. Jackson
Major, United States Army
B.S., South Dakota State University, 1979
M.S., Florida Institute of Technology, 1989

Submitted in partial fulfillment
of the requirements for the degree of

MASTER OF SCIENCE IN PHYSICS

from the

NAVAL POSTGRADUATE SCHOOL
June 1992

Author: Korey V. Jackson

Korey V. Jackson

Approved by: [illegible]

R. F. Kamada, Thesis Advisor

[illegible]

K. E. Woehler, Second Reader

[illegible]

Karlheinz E. Woehler, Chairman
Department of Physics

ABSTRACT

Chaos metrics are examined as a tool to analyze atmospheric three-dimensional dispersion models at the individual particle rather than the aggregate level. These include the self-affine fractal dimension, D_A, Shannon entropy, S, and Lyapunov exponent, λ. Intercomparison of these metrics is first performed with the one-dimensional logistics difference and the two-dimensional Henon systems of equations. The fractal dimension and Shannon entropy are then measured as a function of the inverse Monin-Obukhov length (1/L) for two three-dimensional Lagrangian particle dispersion models, the McNider particle dispersion model and the NPS particle dispersion model now under development. The fractal dimension and Shannon entropy uncover weaknesses in both models which are not obvious with standard geophysical measures. They also reveal similarities and differences between the atmospheric models and simple chaos systems. Combined, these chaos measures may lend detailed insight into the behavior of Lagrangian Monte Carlo dispersion models in general.

TABLE OF CONTENTS

THESIS DISCLAIMER

ACKNOWLEDGEMENTS

I gratefully acknowledge the mentoring, guidance, advice, and experience provided by my thesis advisor, Professor Ray Kamada, in the preparation of this study, and in developing the NPS particle dispersion model on such short order. I also thank Professor Karlheinz Woehler, chairman of the Naval Postgraduate School Physics Department, for serving as my second reader. The support of the United States Army in providing me this opportunity is also appreciated. Finally, I cannot leave unmentioned the support of my wife Tana and daughter Katrina. I sincerely thank all.

I. INTRODUCTION

A. OVERVIEW

Unlike the relatively simple linear relationships studied in basic physics, natural phenomena often exhibit complex nonlinear behavior. One familiar consequence is the inability of meteorologists to make accurate long-term weather forecasts (Devaney, 1990). Like turbulence in other systems, atmospheric weather systems seem to follow patterns which are chaotic and unpredictable in the long term. The behavior of a particle released in such a system is deterministic rather than stochastic. However, small differences in initial conditions result in particle paths which diverge exponentially with time. Thus, since the resolution and accuracy of measurements of the initial conditions are always limited, forecasts will diverge from reality exponentially with time such that long term predictability becomes impossible (Gleich, 1987).

In fluids, predictability relates to particle diffusion. Although the relationship is not simple, less predictable systems tend to more diffusive.

A good Lagrangian particle model which can accurately simulate diffusion over a wide range of atmospheric conditions is of value in predicting toxic dispersion from various sources, such as biochemical warfare agents, Navy LNG tanker leaks, nuclear reactor and weapons plumes, industrial chemical plants, large rocket and missile launch emissions, aborts, and static test firings, and a variety of planned and emergency releases from tanks and hoses involved in storage, transfer, and transport of dispersive volatile toxins. Presently, predictions using existing particle models may vary greatly from reality when operating in certain atmospheric conditions. (Venkatram and Wyngaard, 1988).

Heretofore, atmospheric particle dispersion models have generally been compared against turbulence data in terms of spectra and standard geophysical turbulence measures, such as the turbulent kinetic energy (tke), its component velocity variances (σ_u^2, σ_v^2, and σ_w^2), and the Brunt-Vaisala frequency (BVF), an atmospheric stability measure. The aggregate diffusion of many Lagrangian particles has also been compared with the results of Gaussian plume statistical theories as well as measurements of dispersing clouds (Venkatram and Wyngaard, 1988). However, complex but wholly predictable periodic behavior such as ocean tides may appear quite similar to truly chaotic or random behavior when using such techniques. Another potential set of tools in analyzing particle dispersion model performance are the recently developed chaos metrics such as fractal dimension, D_Λ, Shannon entropy, S, and the Lyapunov exponent, λ (Baker and Gollub, 1990).

B. OBJECTIVE

This study examines the self-affine fractal dimension, D_A, and compares it to the Shannon entropy, S, and the Lyapunov exponent, λ. These metrics are first applied to the 1-D chaotic system known as the logistics difference relation and the chaotic 2-D Henon system. These simple well characterized systems are studied to determine similarities and differences between the above three chaos metrics. Two 3-D Monte Carlo Lagrangian scattering routines designed to simulate atmospheric dispersion are then studied as representative of more complex real world systems.

C. WHY USE CHAOS METRICS

There are at least two good reasons to try chaos metrics on dispersion. One, chaos metrics may provide a means to discriminate between periodic wave behavior, chaos, and differing degrees of turbulence. Second, chaos metrics may provide some additional insight into the behavior of particle dispersion in 3-D scattering routines.

Kamada and DeCaria (1992) have shown that though nocturnal periodic gravity waves have quite different dispersion characteristics from turbulence, the two cannot be distinguished readily by using standard atmospheric turbulence measures such as the Brunt-Vaisala frequency (BVF), vertical velocity variance (σ_w^2), turbulent kinetic energy (tke), buoyancy length scale (l_b), or spectral analysis using FFTs. The self-affine fractal dimension, D_A, was shown to be the only facile wave/turbulence discriminant tested. Since the Shannon entropy and Lyapunov exponent are two other standard chaos measures, they might also be tested as possibly useful dispersion measures.

In the 3-D Monte Carlo scattering routines which model atmospheric dispersion, such as the McNider Particle Dispersion Model, chaos metrics might quickly determine certain model characteristics for a given range of parameters. For example, in examining an expanding cloud of particles, the Lyapunov exponent would measure the divergence rate of the particles. The Shannon entropy would measure the evenness of distribution of particles within the expanding cloud. The self-affine fractal dimension could measure the total apparent distance a particle travels in a time, T, as a function of the time resolution, ϵ, used within T. This might give an indication of the range of scales over which mixing occurs.

II. THEORY

A. OVERVIEW

There are several methods of measuring chaos and turbulence. Of interest for the present purposes are the self-affine fractal dimension, D_A, the Shannon entropy, S, and the Lyapunov exponent, λ. Some standard geophysical measures will also be used for comparison purposes in the 3-D case. Descriptions of each are given below and the logistics difference, Henon, and atmospheric particle diffusion systems to which they are applied are then described.

B. FRACTAL DIMENSION (D_A)

The self-similar fractal dimension, D_B, can be described readily with the Cantor set (Devaney, 1990). To form this set, a straight line is drawn with the middle third removed, as in the second line of Figure 1. The two resulting straight lines, which are one-third the original line length, are then similarly subdivided. The four new lines, all 1/9th the original line length, are further subdivided, *ad infinitum*. From top to bottom in Figure 1, the resulting sets of lines are **self-similar**; that is, the manner in which the geometry is altered is repeated at all successive levels of resolution.

The Koch snowflake is another geometrically self-similar figure (Devaney, 1990). It is constructed by removing the

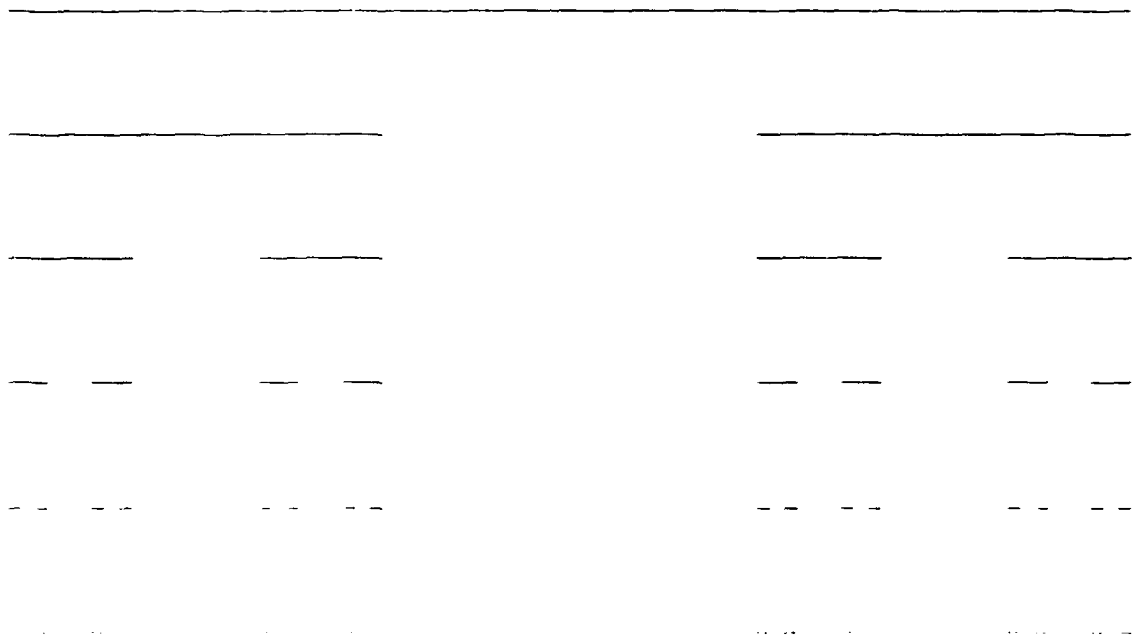

Figure 1, Construction of the Cantor Set.

middle third of each side of an equilateral triangle, and replacing it with two pieces of equal length, creating a six-pointed star. This star has 12 sides, all of length 1/3. The middle third of each of the 12 sides is again removed, and replaced with two lines of length 1/9. Again, the process is repeated *ad infinitum*.

Like the Cantor set, the Koch snowflake is also self-similar [Figure 2]. The jagged sides appear geometrically similar at increasing levels of resolution.

The self-similar fractal dimension is defined by (Devaney, 1990) as

$$D_B = \frac{\ln\,(\mathit{Total\ length\ of\ pieces})}{\ln\,(\mathit{resolution})} \ . \qquad \text{(II-1)}$$

In the Cantor set example, the total length of segments of unit length at level n is 2^n, and the resolution level is 3^n, so that

$$D_B = \frac{\ln 2^n}{\ln 3^n} = \frac{n \ln 2}{n \ln 3} \approx 0.6309 \ . \qquad \text{(II-2)}$$

Similarly for the Koch snowflake, there are 4 pieces for each level of n with a magnification of 3, so that

$$D_B = \frac{\ln 4^n}{\ln 3^n} \approx 1.262 \ . \qquad \text{(II-3)}$$

With geometric figures, D_B is unitless; the measure involves a length divided by a length. However this definition of fractal dimension cannot apply directly to a time series

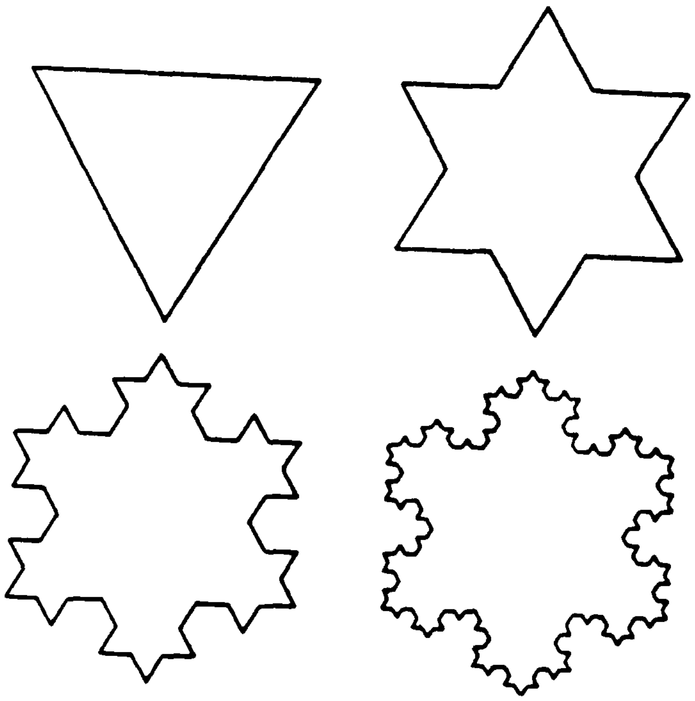

Figure 2. Construction of the Koch Snowflake. From Devaney (1990).

trace, since it would involve the square root of the amplitude squared plus the time squared, i.e.,

$$D_B = \frac{\ln\left[\sum_0^T \sqrt{\Delta A^2 + \Delta t^2}\right]}{\ln \delta t} \quad . \tag{II-4}$$

This definition must be adjusted when applied to time series data to ensure that the end result is independent of the arbitrary unit scaling between amplitude and time.

There are other ways to characterize the fractal dimension of a system. For a more suitable times series measure, McHardy and Czerny (1987) redefined the length metric as

$$L(\epsilon) = \frac{1}{\epsilon}\int_0^T |F(t+\epsilon) - F(t)|\, dt \ . \tag{II-5}$$

where F is the amplitude of the time series at time, t, ϵ is the time increment, and T is the time window over which L is defined.

This definition effectively avoids the units scaling problem. Since the inverse time in $1/\epsilon$ cancels the time units in the integral, $L(\epsilon)$ is only in units of amplitude. The fractal dimension is then defined as

$$D_A \equiv - \frac{d \ln L(\epsilon)}{d \ln \epsilon} \quad . \tag{II-6}$$

McHardy and Czerny applied these definitions to the time variance of X-ray luminosity data from the Seyfert galaxy NG5506. Collins and Kastogi (1989) later applied McHardy and

Czerny's definitions in analyzing gravity wave spectra from the atmospheric mid-troposphere. Recently, Kamada and DeCaria (1992) applied D_A as a tool for discriminating waves from turbulence in nocturnal atmospheric boundary layer data.

To actually compute this function, the integral is approximated with a numerical summation, so that

$$L(\epsilon) = \frac{\delta t}{\epsilon} \sum_{0}^{T} |F(t-\epsilon) - F(t)| \quad . \qquad \text{(II-7)}$$

Since at each resolution $\delta t=\epsilon$, the leading term on the right-hand side is always unity, so that the length is now

$$L(\epsilon) = \sum_{0}^{T} |F(t-\epsilon) - F(t)| \quad . \qquad \text{(II-8)}$$

Thus $L(\epsilon)$ is the total amplitude change over a time series of length, T, for a given time resolution, ϵ. In this form it is quite clear that time is removed from the length determination, so that $L(\epsilon)$ does not depend on some arbitrary scaling between amplitude and time.

C. SHANNON ENTROPY (S)

1. Definition

When a particle is first released, its initial location is known and completely specified, so the information entropy (defined later) for its location is zero. Later, according to given equations of motion, its position diverges from the initial point. If the range of its possible positions is partitioned into equal segments, then for a given time interval, T, its motion can be recorded in terms of occupancy time for each segment. Then the particle dispersion rate might be measured by the seeming degree of randomness of occupancy time or evenness of the state probability distribution over time period, T. Shannon or information entropy is defined by the state probability distribution, so entropy may be regarded as a measure of dispersion rate for a time series of particle states. This can apply to the actual particle position, its velocity, or its phase velocity. Shannon entropy is defined simply as (Baker and Gollub, 1990)

$$S = -\sum_{i=1}^{N} p_i \ln p_i \ , \qquad \text{(II-9)}$$

where

$S \equiv$ *system entropy,*

$N \equiv$ *number of permitted states, and*

$p_i \equiv$ *probability of state i, such that* $\sum_{i=1}^{N} p_i = 1.0$.

Then, for N permitted states (or position intervals), the maximum possible entropy corresponds to equal occupancy time in each of the N states, so p_i is 1/N for all i. That is,

$$S_{\max} = -\sum_{i=1}^{N} \frac{1}{N} \ln \frac{1}{N} \ . \qquad \text{(II-10a)}$$

Expanding the summation results in

$$S_{\max} = \frac{1}{N}\ln\frac{1}{N} + \frac{1}{N}\ln\frac{1}{N} + \frac{1}{N}\ln\frac{1}{N} + \ldots \quad , \qquad \text{(II-10b)}$$

and collapsing the common multiples gives

$$S_{\max} = -N\left(\frac{1}{N}\ln\frac{1}{N}\right) \quad ,$$
$$S_{\max} = \ln N \quad , \qquad \text{(II-10c)}$$

which corresponds to total randomness, or a completely even particle distribution across all allowable states. Also,

$$S_{\min} = 0 \quad , \qquad \text{(II-11)}$$

which corresponds to all particles being in one state.

For a randomly moving particle in 2-D space, the computation is similar. The 2-D space may be divided into, say, a 100 X 100 grid. If the particle is moving completely randomly, at time, t, it may be in any one of the grid boxes with equal probability. The entropy for this system is then

$$S = -\left[\left(\frac{1}{100^2}\ln\frac{1}{100^2}\right) + \left(\frac{1}{100^2}\ln\frac{1}{100^2}\right) + \ldots\right] \quad ,$$
$$S = -\,100^2\left(\frac{1}{100^2}\ln\frac{1}{100^2}\right) \quad ,$$

(II-12a)

which is the same as

$$S = \ln 100^2 = S_{\max} \quad . \qquad \text{(II-12b)}$$

2. Shannon Entropy for an N-D System

For a 1-D system, the domain is simply partitioned into n intervals, and the probability computed for each interval. For a simple system such as recursion of the logistics difference equation,

$$x_{n+1} = \mu x_n (1 - x_n) \quad , \qquad \text{(II-13)}$$

the probability for the ith interval, p_i, is simply the number of times the interval has been occupied after n number of recursion steps, divided by the total number of steps.

Note that the number of partitions should be appropriate for the total number of steps. Having too few intervals is equivalent to too low a resolution of the entropy and may result in a misleadingly low value of S. For instance, with only one interval, $p_i = 1$, and $S = 0$. Ideally, the number of intervals for maximum resolution of the entropy of the system is $e^{\lambda N}$, where λ is the positive Lyapunov exponent for the system and N is the number of steps (Baker and Gollub, 1990, pp.126-129). Since λ is typically of order unity, $e^{\lambda N}$ can easily be a computationally intractable number.

Again, for 3-D systems, assuming N equal segments along each axis, the number of partitions will be N^3, so computation quickly becomes unwieldy for large N. In practice, N of order 10^2 to 10^3 has been used by some authors (Baker and Gollub, 1990, pg. 88) as a compromise between entropy resolution and computational efficiency. In analyzing real data, the number of partitions should be related in some direct fashion to the maximum resolution, ϵ_i, of the measuring devices, and the total measurement time, T. The total number of partitions should probably not exceed T/ϵ_i, or otherwise even a completely random distribution would still result in some intervals unoccupied.

3. Ln vs Log$_2$

Wolf maintains that the Shannon entropy should be computed with $\log_2$ rather than the natural logarithm, $\log_e$ (Wolf, 1986, pg. 276). With $\log_2$, the Shannon entropy relates directly to information in the form of binary bits, since one bit of information can be specified as being in only one of two states, e.g., true or false, on or off, one or zero. That is, the $\log_2$ basis sets the entropy equal to the minimal length of binary code required to describe the state of the system. If all particles occupy only one state, turning the bit for that state to the "on" position is sufficient to specify the state of the system. If the particles are evenly distributed among all states, the length of binary code required to specify the system is equal to the number of states, which means that the system is completely random (Tribus and McIrvine, 1971). However, for the purposes of this study, the Shannon entropy can also be written in the following form:

$$S = -K\sum_{i=1}^{N} p_i \ln p_i \quad . \qquad \text{(II-14)}$$

The only difference in measuring entropy by the natural logarithm versus base two is the value imposed on the constant K. Most computations assume that K=1. As long as the method of computing S is similar when making comparisons of computed entropy, there need be no confusion.

D. LYAPUNOV EXPONENT (λ)

For an expanding turbulent cloud of particles, one measure to describe the system is the particle divergence rate. Due to turbulence, the trajectories of two particles arbitrarily close will diverge with time. The divergence rate can be characterized by Lyapunov exponents. The Lyapunov exponent, λ, is also a measure of the system's sensitivity to initial conditions. In an N-dimensional system, there will be N Lyapunov exponents; however, these do not necessarily correspond to coordinate axes. So in a Cartesian coordinate system, λ_1, λ_2, and λ_3 are locally defined Lyapunov exponents which generally cannot be described as λ_x, λ_y, and λ_z. Direction is adjusted for each point along the trajectory. In one-dimension, the Lyapunov exponent is defined by (Wolf, 1986, p. 275) as

$$\lambda = \lim_{N \to \infty} \frac{1}{N} \sum_{i=0}^{N-1} \ln |f'(i)| \quad , \qquad \text{(II-15)}$$

which can be described over a map domain as an integral,

$$\lambda = \int_{0}^{N} p_i \ln |f'(i)| \, di \quad . \qquad \text{(II-16)}$$

Overall, the Lyapunov exponent can be viewed as a measure of the average local stretching rate of the particle trajectory, as indicated by the log of the length of the slope, $|f'(i)|$, weighted by the probability of encountering that slope. (Wolf, 1986)

The Lyapunov exponent is related to the entropy, S, as well as the information loss rate. For instance, if a measured position is presently known to a precision of 16 bits, and λ = 2 bits per second, then the particle's future trajectory cannot be predicted to any degree of precision beyond 8 seconds (16 orbits).

It should be noted that the Lyapunov exponent defines the **average** rate of loss of predictive power, and may vary locally along the orbit.

The Lyapunov exponent is readily computed for one dimension, but in higher dimensions calculation becomes more complex. In three dimensions, the directions of trajectory divergence and contraction must be defined in terms of local tangents, which vary from point to point, so the calculation of the exponents must constantly adjust for each change in direction. Wolf has developed algorithms for computing higher dimensional Lyapunov exponents which involve reorienting the major axis of an ellipse for each point, then renormalizing the function after every few points so that the unit ellipsoid does not overlap an attractor (Wolf, 1986).

In 3-D, three Lyapunov exponents are required to classify the system. A negative exponent indicates a dissipative dimension. If all three exponents are negative, the system is dissipative, e.g., a pendulum settling down to a fixed point. If an exponent is zero, the system orbits about a fixed point. If one of the three exponents is positive, the system is chaotic; the orbital trajectories are diverging.

E. GEOPHYSICAL MEASURES OF TURBULENCE

1. Overview

Geophysicists use several standard methods to describe turbulence. Discussed here are those methods used to examine the McNider and NPS dispersion models.

2. Turbulent Kinetic Energy (tke)

The mean turbulent kinetic energy, or TKE, is defined as

$$tke \equiv \frac{1}{2}(\sigma_u^2 + \sigma_v^2 + \sigma_w^2) \quad , \qquad \text{(II-17)}$$

where

σ_u ≡ standard deviation, u component of particle velocity,
σ_v ≡ standard deviation, v component of particle velocity,
σ_w ≡ standard deviation, w component of particle velocity.

3. Gradient Richardson Number (Ri)

The gradient Richardson number is defined by

$$Ri \equiv \frac{g}{\theta} \frac{\frac{\partial\overline{\theta}}{\partial z}}{\left[\left(\frac{\partial\overline{u}}{\partial z}\right)^2 + \left(\frac{\partial\overline{v}}{\partial z}\right)^2\right]} , \qquad \text{(II-18)}$$

where the overbars signify time averages, and

g ≡ earth's gravitational acceleration,
z ≡ vertical position above the surface,
θ ≡ potential temperature, i.e. the temperature normalized for adiabatic expansion with pressure, so

$$\theta \equiv T\left(\frac{p}{p_0}\right)^{\frac{R_d}{c_p}} , \qquad \text{(II-19)}$$

where R_d is the gas constant for dry air, and C_p is the specific heat at constant pressure.

Ri is used in place of the Reynolds number as a dynamic indicator of turbulence when the atmosphere displays a non-neutral density profile. The numerator is proportional to the buoyant production or destruction rate of tke, depending on negative or positive sign, respectively. The denominator is proportional to the shear generation rate of tke, which is nearly always positive. Thus, a positive Richardson number indicates a stable atmosphere (increasing potential temperature with height) and suppression of turbulence. Commonly, when the buoyancy destruction rate of tke exceeds 1/4 the shear production rate, turbulence is suppressed, i.e., when Ri > 1/4 (Stull, 1988, pg. 176).

4. Brunt-Vaisala Frequency (BVF)

The Brunt-Vaisala frequency, BVF, is a measure of static stability. BVF is defined by

$$BVF \equiv \sqrt{\frac{g}{\overline{\theta}_v}\frac{\partial\overline{\theta}_v}{\partial z}} \quad , \qquad \text{(II-20)}$$

and in principal gives the highest gravity wave frequency which a fluid can support. It is undefined for negative temperature gradients, i.e., an unstable atmosphere. (Sorbjan, 1989, pg. 35).

5. Buoyancy Length (l_B)

The buoyancy length, l_b, is the standard deviation of the vertical velocity divided by the Brunt-Vaisala frequency,

$$l_B \equiv \frac{\sigma_w}{BVF} \quad . \qquad \text{(II-21)}$$

The buoyancy length is meant to be a measure of the dominant eddy scale. (Stull, 1988, pg. 310).

6. Monin-Obukhov Length (L)

The Monin-Obukhov length is given by

$$L \equiv -\frac{u_*^3\,\theta}{g\,k\,\overline{w'\theta'_0}} \quad , \qquad \text{(II-22)}$$

where

$\overline{w'\theta'_0} \equiv$ *surface temperature flux* ,
$k \equiv$ *von Karman constant* ≈ 0.4 ,

$$\text{and} \quad u_* = \frac{V}{\sqrt{\ln\frac{z}{z_0} - \psi_m}} \quad . \qquad \text{(II-23)}$$

u. is the friction velocity, a measure of surface drag due to turbulent friction. The Businger function, ψ_m, is a stability correction which is approximated with a rational function by (Kamada, 1992b)

$$\psi_m = \begin{cases} -5\frac{z}{L} , & \frac{z}{L} > 0\,(stable) , \\ \dfrac{0.15961583 - 5.4151107\frac{z}{L}}{1 - 3.59002232\frac{z}{L} - 0.799168457\left(\frac{z}{L}\right)^2} , & \frac{z}{L} \le 0\,(unstable) . \end{cases}$$

(II-24)

In the convective boundary layer, L is proportional to the height at which the buoyant generation rate of tke matches the shear generation rate. L is the primary stability measure for the atmospheric surface layer and is also used in combination with the inversion height, z_i, to characterize boundary layer stability. L > 0 indicates a stable boundary layer where vertical turbulence is suppressed by positive density gradients with height, z. L < 0 indicates an unstable surface layer where upward vertical motion is encouraged by negative density gradients. A stable atmosphere implies that a vertically displaced air parcel will tend to return to its original height. (Businger, 1973).

F. 1-D AND 2-D CHAOS EQUATIONS. LOGISTICS & HENON FUNCTIONS

1. Overview

The initial focus of this study is to compare and test some simple potential dispersion metrics. The fractal dimension, D_A, the Shannon entropy, S, and the Lyapunov exponent, λ, are simple expressions which can monitor transitions between periodic and chaotic behavior, akin to the transition between laminar and turbulent behavior in a real fluid. Two such expressions, well characterized in the literature, are the logistics difference equation:

$$x_{n+1}=\mu x_n(1-x_n) \quad , \tag{II-25}$$

and the Henon system:

$$\begin{aligned} x_{n+1} &= 1 - ax_n^2 + y_n \quad , \\ y_{n+1} &= bx_n \quad . \end{aligned} \tag{II-26}$$

(Gould and Tobochnik, 1988, pp.152-178; Baker and Gollub, 1990).

2. The Logistics Difference Equation

Though simple and one dimensional, recursive iteration of the logistics difference equation results in chaotic behavior for certain parameters. For $0 < x_0 < 1$ and $\mu < 1$, x_n converges to 0. For $1 < \mu < 3$, and the same range for x_0, x_n

converges to $\mu/4$. For $3 < \mu < 3.6$, x_n fluctuates between 2^n discrete points; while beyond $\mu \simeq 3.6$, the solution is nearly random, i.e., chaotic. [Figure 3]

3. **Graphical Analysis of the Logistics Difference Equation**

The logistics difference equation is parabolic, so that for an initial x_0 (given as 0.04 with μ=2.9 in Figure 4), drawing a vertical line to the parabola corresponds to x_1. Then a horizontal line drawn to the inscribed straight line of slope unity corresponds to recursing x_{n+1} to a new value for x_n. Reiterating this procedure marches the solutions to a final value of 0.655.

This final value can be determined exactly from the original function. At steady state, $x_{n+1} = x_n$. Then for Figure 4, where $\mu = 2.9$,

$$x_{n+1} = 2.9 x_n (1 - x_n) \ , \qquad \text{(II-25)}$$

and solving for x_n:

$$x_n = 0 , \quad x_n = 0.65517 \ . \qquad \text{(II-26)}$$

Note since the equation is quadratic, that there are two solutions. The x_n=0 solution is not stable (as defined below). The general rule is: if the magnitude of the slope of the function in the region of the solution is greater than 45°, the fixed point is an unstable solution; if the slope is less than 45°, the fixed point is a stable solution.

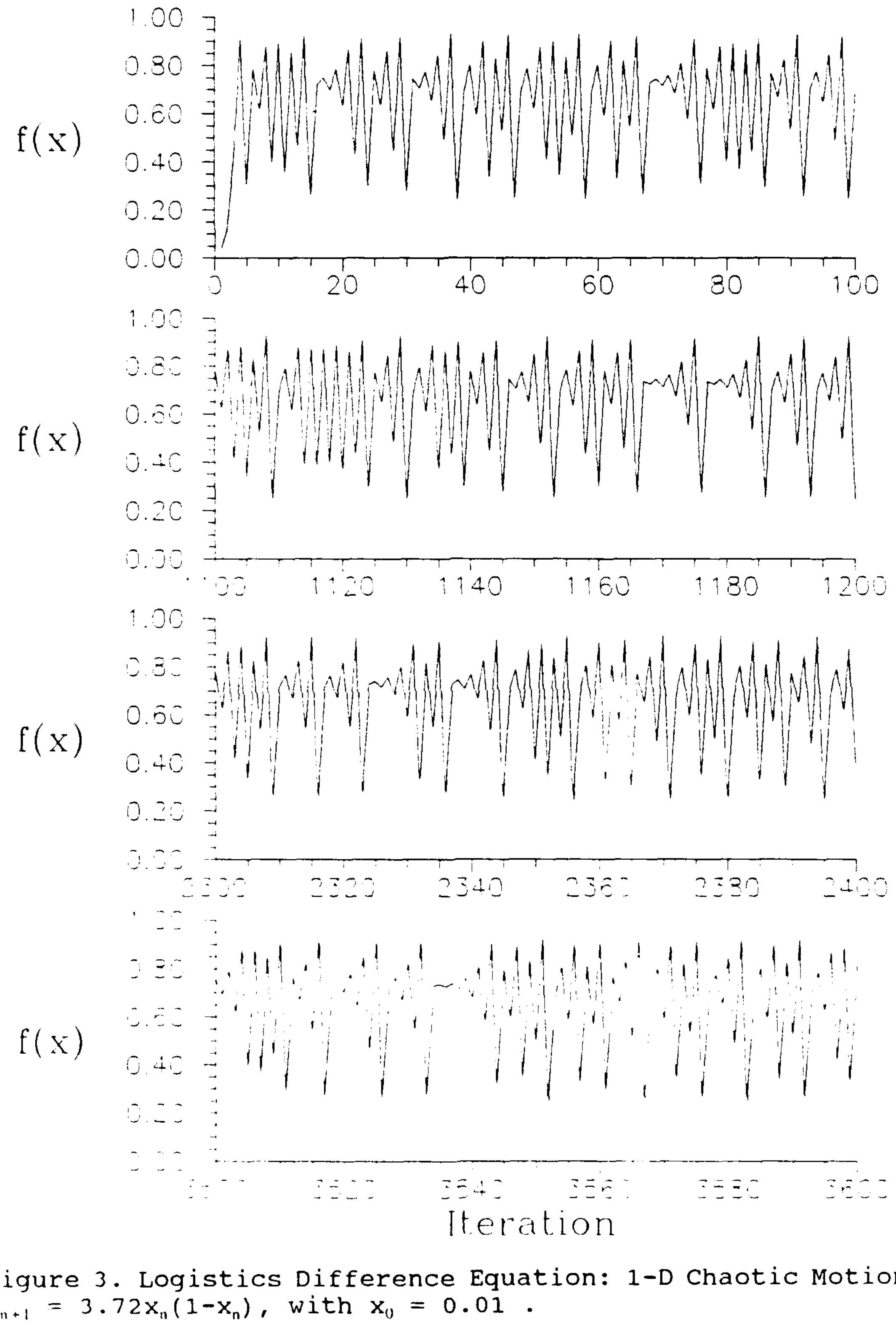

Figure 3. Logistics Difference Equation: 1-D Chaotic Motion. $x_{n+1} = 3.72x_n(1-x_n)$, with $x_0 = 0.01$.

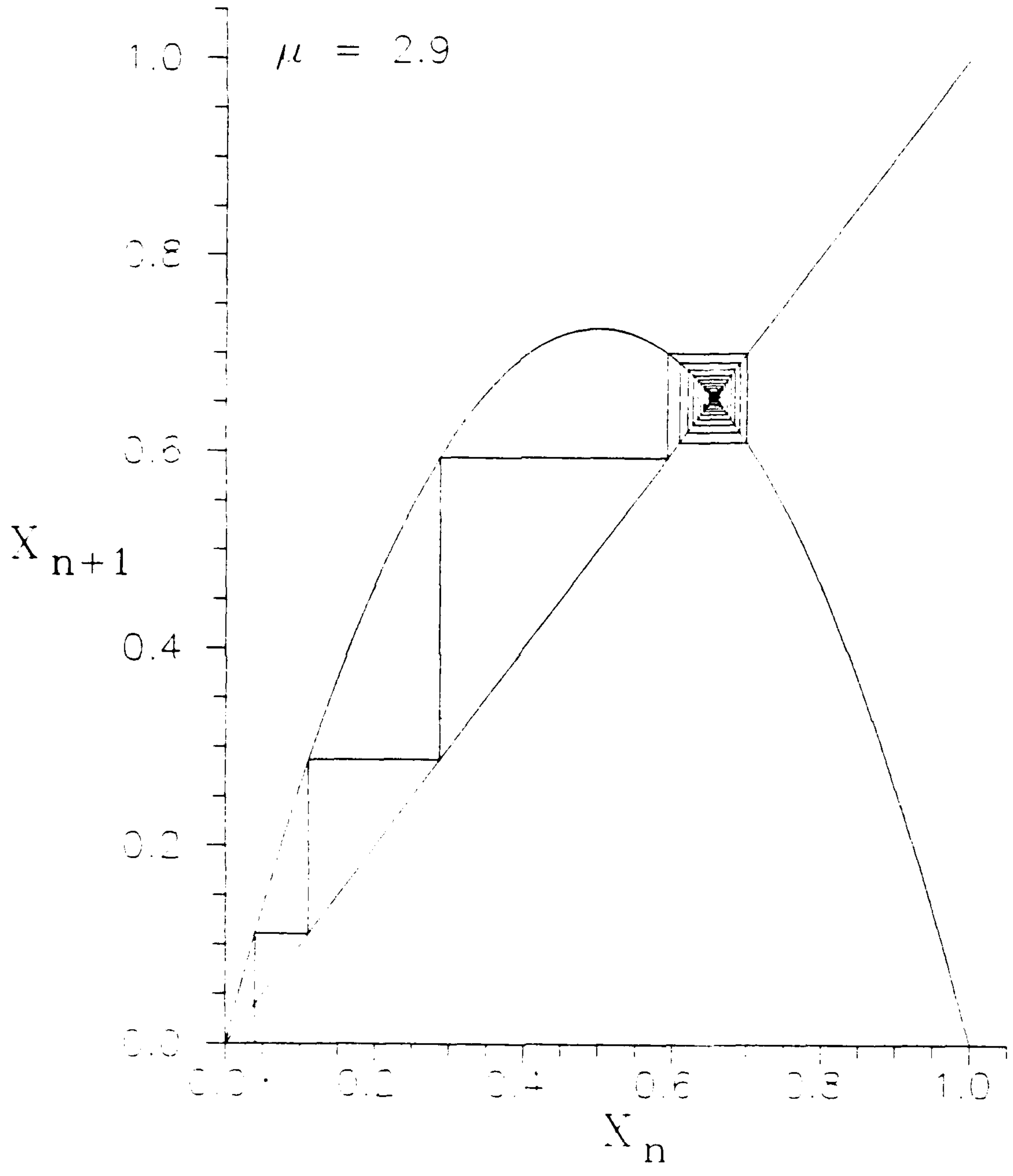

Figure 4. Logistics Difference Equation Graphical Analysis. $x_{n+1} = 2.9x_n(1-x_n)$.

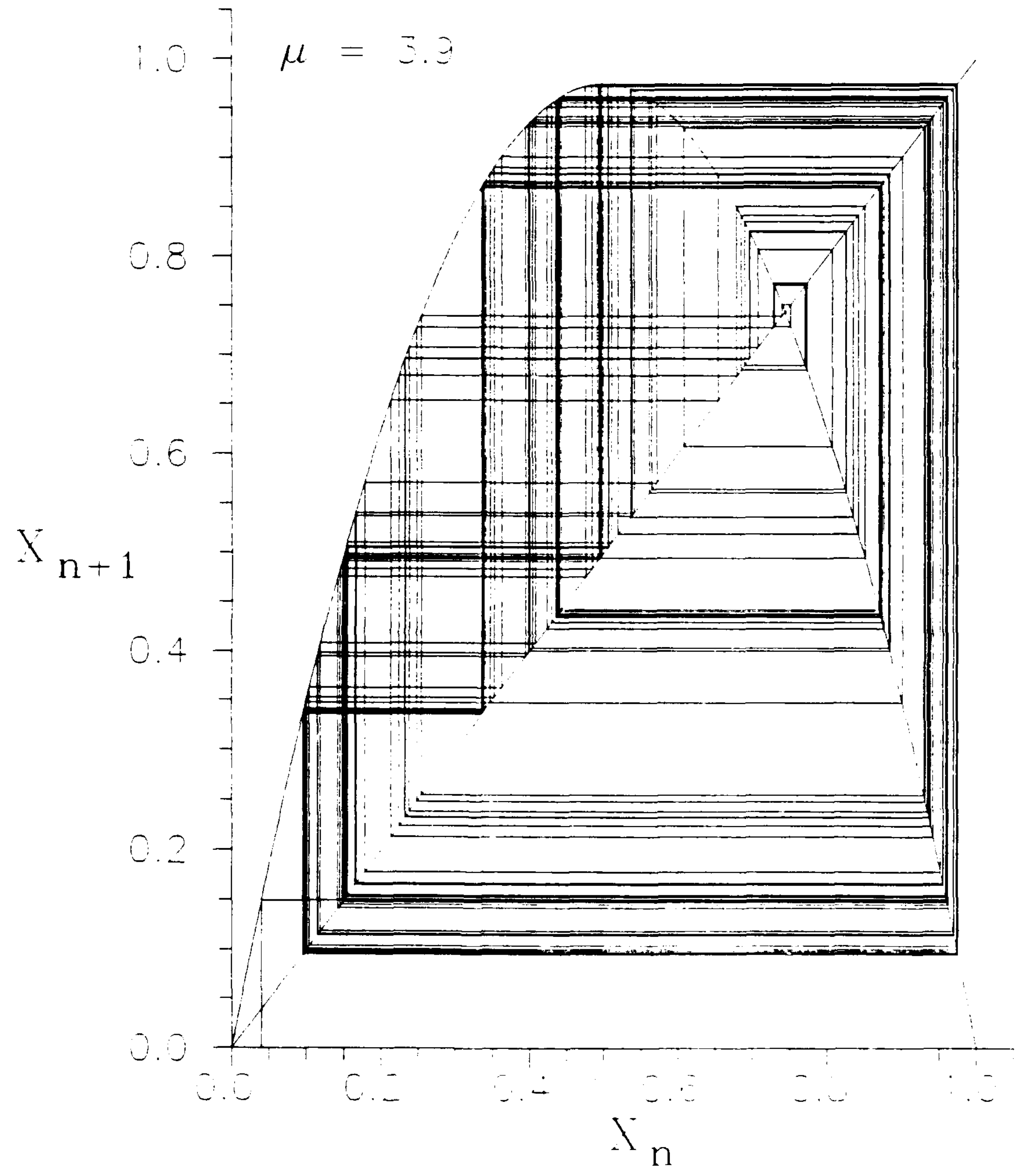

Figure 5. Logistics Difference Equation Graphical Analysis. $x_{n+1} = 3.9x_n(1-x_n)$

When the function is chaotic, there are still only two solutions to the quadratic equation. However, both points are unstable: they both repel rather than attract the value of x_{n+1}. The two points can still be computed, resulting in

$$x_n = 0\,, \quad x_n = 0.74 \quad . \qquad \text{(II-27)}$$

Another way of looking at this is by taking the derivative of x_{n+1} with respect to x_n:

$$\frac{d\,x_{n+1}}{d\,x_n} = \mu\,(1 - 2x_n) \quad . \qquad \text{(II-28)}$$

Checking the chaotic system of Figure 5, when $x_n = 0.74$, the slope is

$$\frac{dx_{n+1}}{dx_n} = 3.9(1 - 2(0.74)) = -1.87 \quad , \qquad \text{(II-29)}$$

which is less than -1, and hence unstable.

This concept is vital when visualizing what the Lyapunov exponent is measuring. Any time both solutions are at points where the slope is greater than 45° or less than -45°, the Lyapunov exponent is greater than zero (ln of the slope, which is greater than unity), and the system is chaotic.

4. Bifurcation Diagrams

A map of the possible values of x_n for various μ is called a bifurcation diagram [Figure 6]. With the bifurcation diagram, the complete behavior of the function can be determined. For example, for $0 \leq \mu \leq 3$, x_n tends to only one

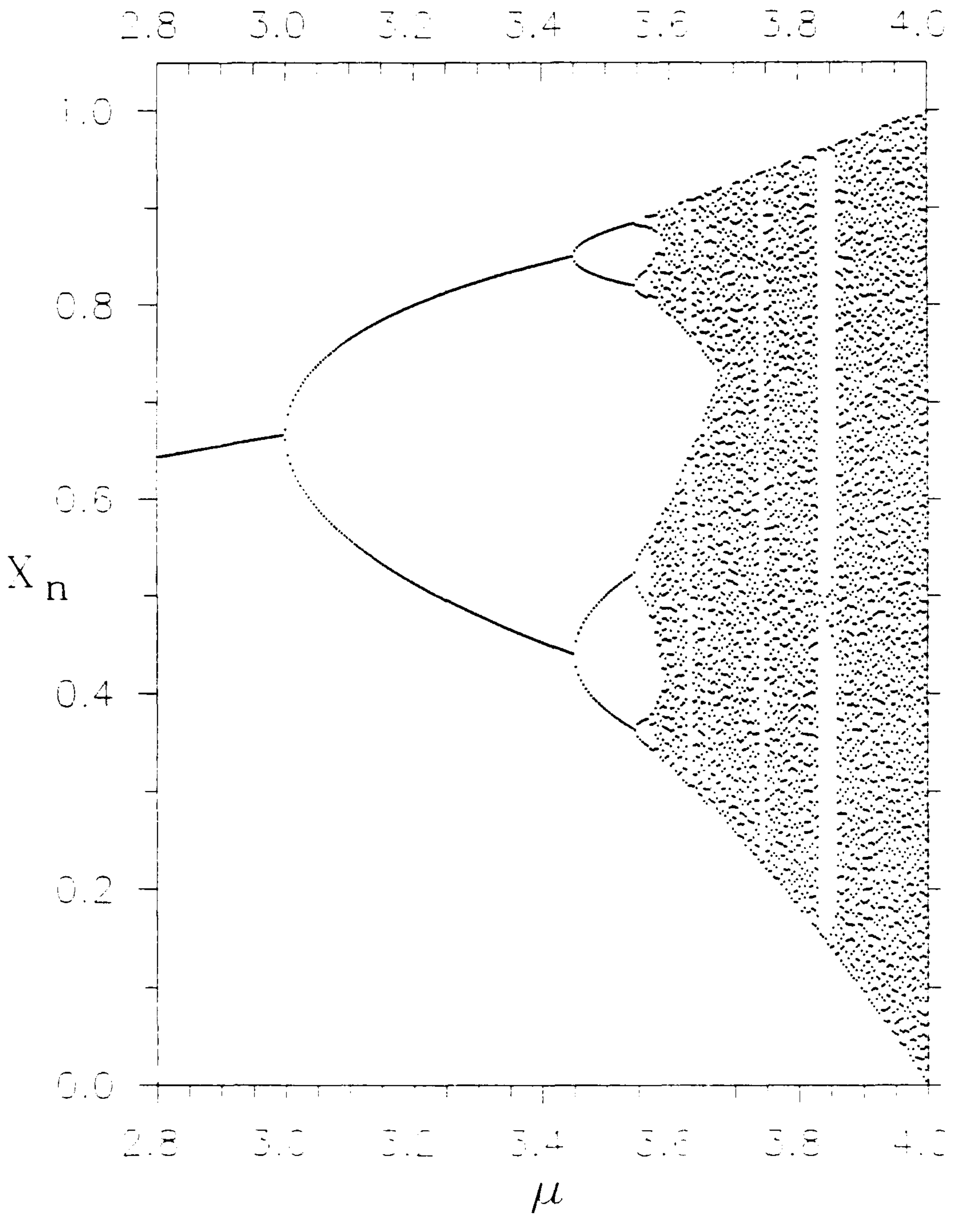

Figure 6. Logistics Difference Equation Bifurcation Diagram. $x_{n+1} = \mu x_n(1-x_n)$.

value; the repetitive recursion of x_{n+1} to x_n points to one attracting fixed point, corresponding to a dissipative system. At $\mu \geq 3$, the bifurcation diagram splits into two fixed points. The solution of x_{n+1} bounces back and forth between these two points. This is period doubling to period 2. Note for still larger values of m that there is another split to period 4, then period 8, followed by a rapid series of bifurcations culminating in chaos, where the points appear to be distributed over a wide range of x_n.

Another curiosity appears in the bifurcation diagram. There are numerous "windows" within the chaotic region. The most obvious is around $\mu \approx 3.82$, where the period 4 and 8 behavior seems to appear again. Increased resolution would show many more such windows. Also note that period widths get progressively narrower, and that for higher periods it is difficult to distinguish say period 64 from chaos.

5. The Henon Equations

As with the logistics difference equation, the Henon set has stable solutions for a range of parameters a and b which correspond to one fixed point. Other values for a and b result in two fixed points, or period doubling [Figure 7], and period 4 and higher. For still higher values of a and b, the result is chaotic behavior: the possible positions appear to be spread throughout a discrete range of x and y [Figure 8].

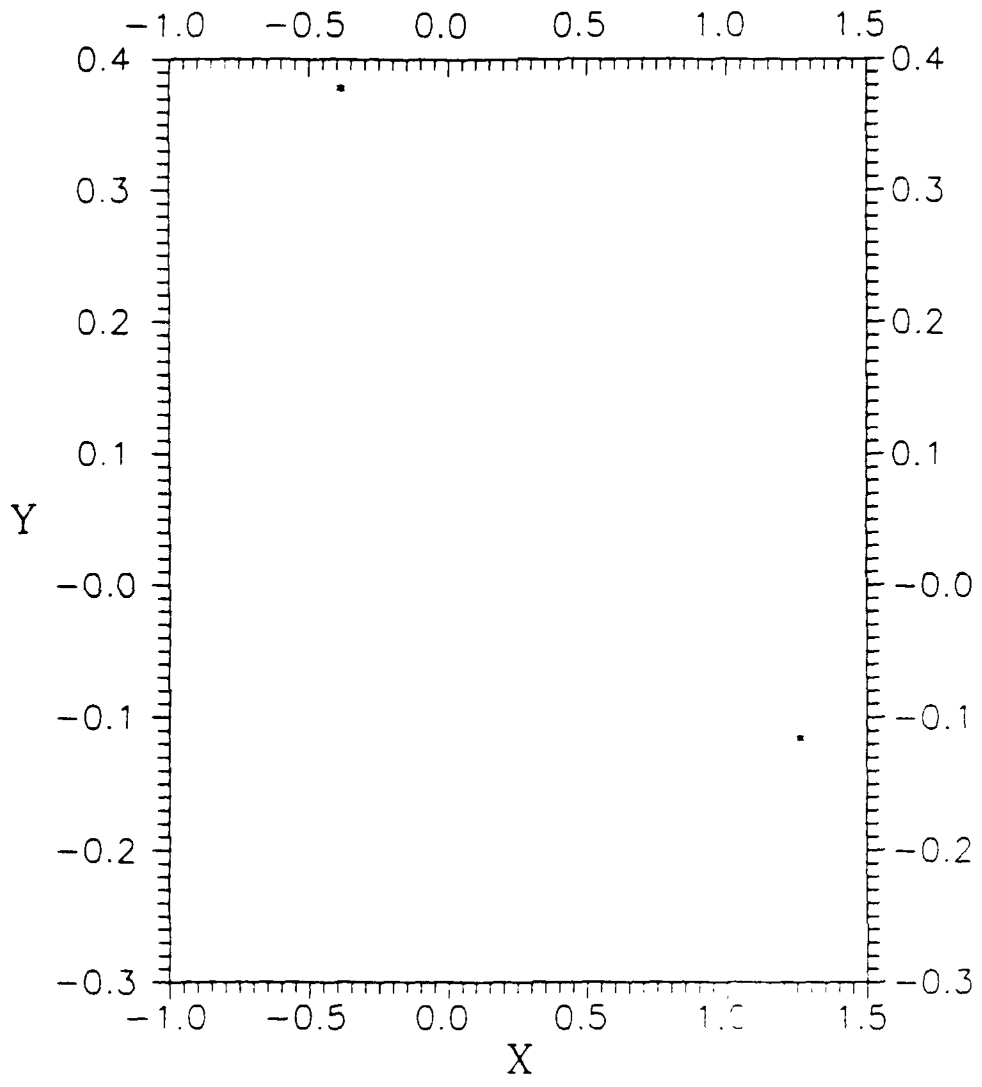

Figure 7. Henon System: Period 2, $x_{n+1} = 1-0.8x_n^2+y_n$, $y_{n+1} = 0.3x_n$.

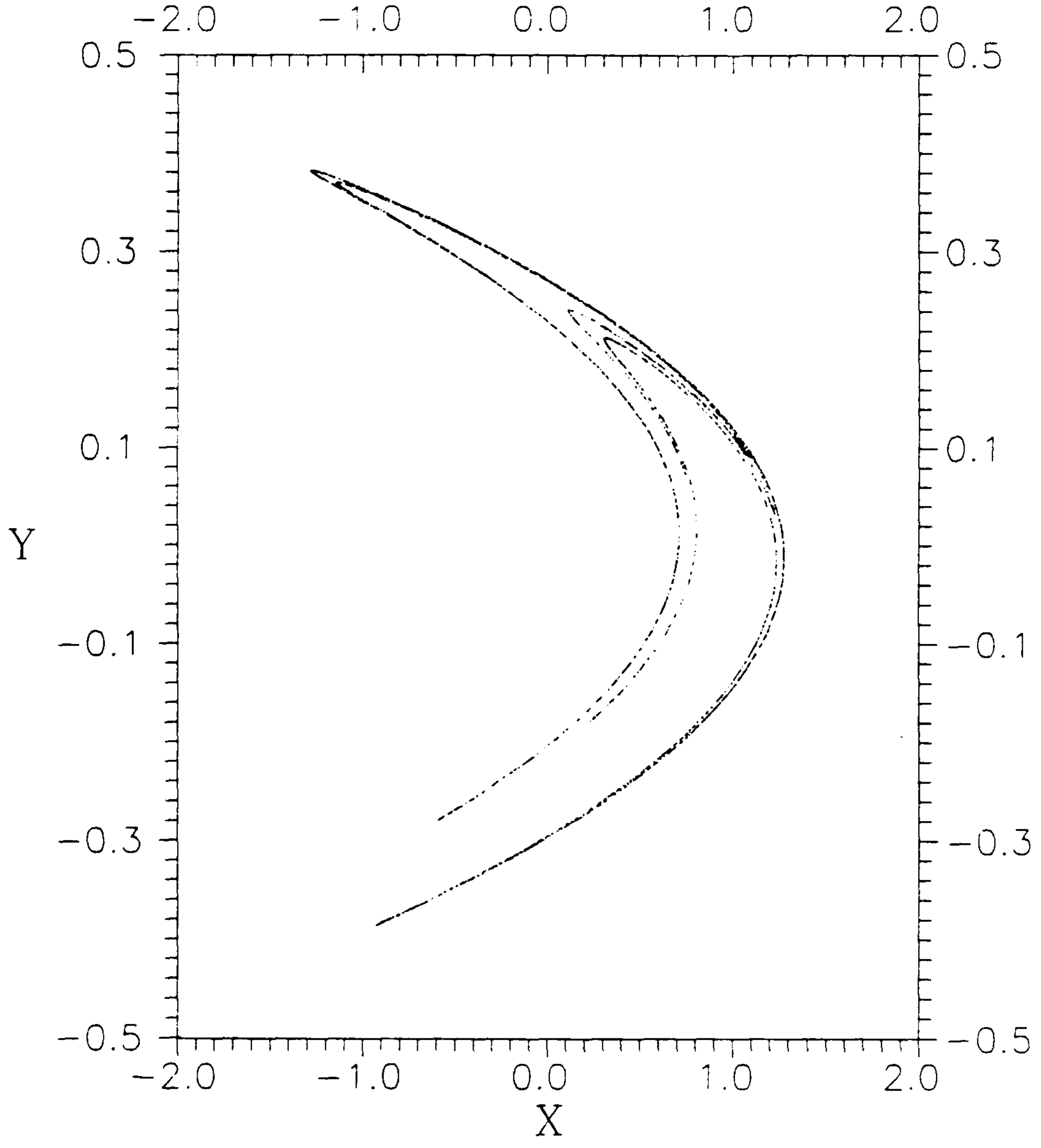

Figure 8. Henon System: Chaos, $x_{n+1} = 1-1.4x_n^2+y_n$, $y_{n+1} = 0.3x_n$.

The Henon bifurcation diagram resembles the logistics difference equation diagram, but has some obvious differences [Figure 9]. One, it does not show symmetric splitting, but rather is skewed downward from the centerline. Two, at the value, $a = 1.08$ (with $b = 0.3$), a window appears in the bifurcation map with new values for x_n; the periodicity in this region occurs at points outside the previous range of x_n. These new points seem to appear out of nowhere, and are not linked to previous points by bifurcation.

G. 3-D ATMOSPHERIC PARTICLE DISPERSION MODELS

1. Overview

The second focus of this study is to examine chaos metrics as performance measures for 3-D Monte Carlo scattering routines. Of immediate interest is particle dispersion within the atmosphere. The McNider Dispersion Model is commonly used to estimate the transport and diffusion of atmospheric pollutants. The model estimates diffusion based on atmospheric flow predictions, and simulates the behavior of pollutant clouds by representing them as the ensemble trajectories of numerous Lagrangian point particles (Pielke, 1984). For this study, the entire range of possible atmospheric stabilities was studied, from $-0.2 \leq 1/L$ 0.1. Therefore, to render the computations tractable, the flow predictions were obtained from boundary similarity theory developed by Sorbjan (1990) and Kamada (1992 a,b) rather than from a mesoscale windflow

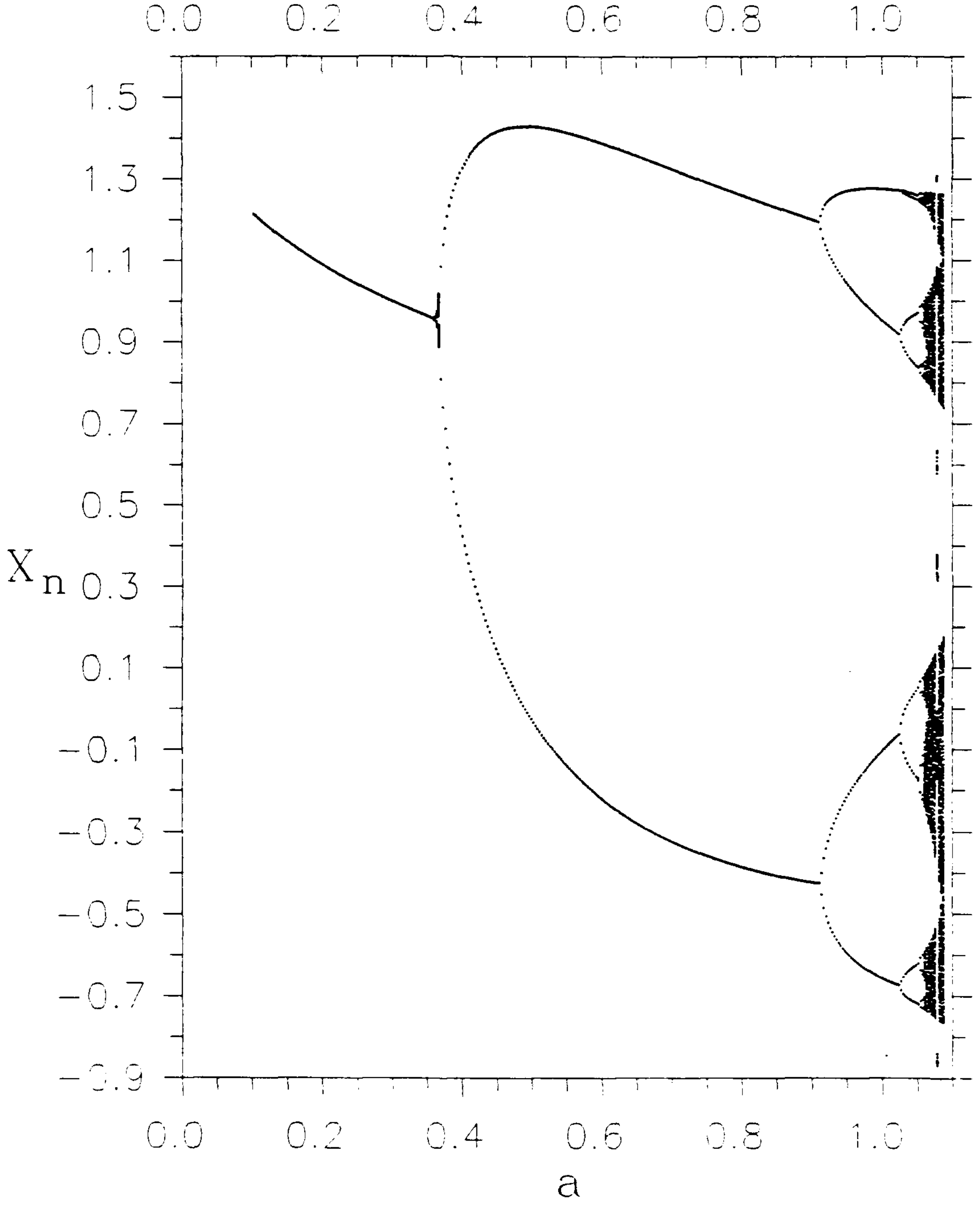

Figure 9. Henon System: Bifurcation Diagram, $x_{n+1} = 1-ax_n^2+y_n$, $y_{n+1} = 0.3x_n$

model, since steady state mean flows could be assumed. The difference is not large, since most of the similarity algorithms are also used in the turbulence closure scheme for the windflow model.

Several weaknesses were perceived in the McNider formulation. Therefore, a similar model was developed at NPS, and was also tested. It features a double Gaussian vertical velocity skewness scheme and damping of the particle oscillations within the boundary layer (Kamada 1992b).

2. The McNider Dispersion Model

For this study, mean flow components are neglected in order to focus on the turbulent fluctuations. The McNider Dispersion Model utilizes the form

$$\begin{aligned} x(t+\Delta t) &= x(t) + u(t)\Delta t, \\ y(t+\Delta t) &= y(t) + v(t)\Delta t, \\ z(t+\Delta t) &= z(t) + w(t)\Delta t, \end{aligned} \qquad \text{(II-30)}$$

where x, y, and z define the particle position in a Cartesian coordinate system, and the u, v, and w terms refer to the fluctuating turbulent velocity components, respectively. The velocity components are computed by (Pielke, 1984)

$$\begin{aligned} u(t) &= u(t-\Delta t)R_u(\Delta t) + u'(t-\Delta t), \\ v(t) &= v(t-\Delta t)R_v(\Delta t) + v'(t-\Delta t), \\ w(t) &= w(t-\Delta t)R_w(\Delta t) + w'(t-\Delta t), \end{aligned} \qquad \text{(II-31)}$$

where R refers to the Lagrangian autocorrelation factors for each velocity component and is a function of Δt. The second terms on the right refer to random fluctuating components of the velocity, and are related to the flow field's turbulent kinetic energy. This random fluctuation of the velocity is constrained by the fluid's turbulent kinetic energy with

$$\begin{aligned} \sigma'_u &= \sigma_u\sqrt{1 - R_u^2(\Delta t)}\,, \\ \sigma'_v &= \sigma_v\sqrt{1 - R_v^2(\Delta t)}\,, \\ \sigma'_w &= \sigma_w\sqrt{1 - R_w^2(\Delta t)}\,, \end{aligned} \qquad \text{(II-32)}$$

where the σ terms refer to the standard deviations of the velocity components. These parameters are not directly computed from atmospheric windflow in McNider's model. Instead an empirical formula is used to deduce these values:

$$\sigma_w = \begin{cases} \dfrac{K_m}{A\lambda_{m_w}}, & \dfrac{z}{L} \le 0\ , \\ 1.2l\left(\dfrac{Ri_c - Ri}{Ri_c}\right)^{0.58}\sqrt{\left(\dfrac{\partial\bar{u}}{\partial z}\right)^2 + \left(\dfrac{\partial\bar{v}}{\partial z}\right)^2}, & \dfrac{z}{L} > 0\ , \end{cases}$$

(II-33)

where

$$l = \begin{cases} 0.35z, & z < 205m\ , \\ 70m, & z \ge 205m\ . \end{cases} \qquad \text{(II-34)}$$

K_m is the local exchange coefficient, described later in the NPS model discussion. To account for advection across vertical gradients of σ_w,

$$\sigma_w = \delta t \frac{\partial \sigma_w}{\partial z} w(t - \Delta t) + \sigma_{w_0} \quad , \qquad \text{(II-35)}$$

and the horizontal components are determined by

$$\sigma_u = \sigma_v = \begin{cases} u_*\left(12 + \dfrac{z_i}{2|L|}\right)^{\frac{1}{3}} , & \dfrac{z}{L} \le 0 \ , \\ 2.3\,u_* \ , & \dfrac{z}{L} > 0 \ . \end{cases} \qquad \text{(II-36)}$$

The maximum wavelength in the vertical velocity spectra is given by

$$\lambda_{m_w} = \begin{cases} \dfrac{z}{\left(0.55 + 0.38\dfrac{z}{L}\right)} \ , & 0 \le z \le |L| \ , \\ 5.9\,z \ , & L < z \le 0.1\,z_i \ , \\ 1.8\,z_i\left[1 - \exp\left(\dfrac{-4z}{z_i}\right) - 0.0003\exp\left(\dfrac{8z}{z_i}\right)\right] , & 0.1\,z_i < z < z_i \ . \end{cases}$$

(II-37a)

for z/L < 0, and

$$\lambda_{m_w} = z \ ; \quad \lambda_{m_w} \le 2.9\,l \ , \quad \frac{z}{L} \ge 0 \ . \qquad \text{(II-37b)}$$

In the above, z_i denotes the top of the planetary boundary layer.

The parameter, Ri, is the gradient Richardson number, defined previously by equation II-19. The critical Richardson number, Ri_C, beyond which turbulence is suppressed, is given by

$$Ri_C = 0.115\ \Delta z^{0.175} \ , \qquad \text{(II-38)}$$

where Δz is measured in centimeters. The Lagrangian autocorrelation terms are determined by

$$\begin{aligned} R_u(\Delta t) &= \exp\left(\frac{-\Delta t}{T_{L_u}}\right) , \\ R_v(\Delta t) &= \exp\left(\frac{-\Delta t}{T_{L_v}}\right) , \\ R_w(\Delta t) &= \exp\left(\frac{-\Delta t}{T_{L_w}}\right) . \end{aligned} \qquad \text{(II-39)}$$

T_L is the Lagrangian integral or e-folding time scale for velocity autocorrelations, and is determined for each component from the turbulence spectra with

$$\begin{aligned} T_{L_u} &= 0.2\,\beta_u \lambda_{m_u} / \overline{V} , \\ T_{L_v} &= 0.2\,\beta_v \lambda_{m_v} / \overline{V} , \\ T_{L_w} &= 0.2\,\beta_w \lambda_{m_w} / \overline{V} , \end{aligned} \qquad \text{(II-40)}$$

where λ_m is the dominant wavelength for each component. Furthermore, the average velocity is defined by

$$\overline{V} = \sqrt{\overline{u^2} + \overline{v^2} + \overline{w^2}} \ . \qquad \text{(II-41)}$$

β, the ratio of Lagrangian to Eulerian time scales, is given by

$$\beta_u = 0.6\frac{\overline{V}}{\sigma_u},$$
$$\beta_v = 0.6\frac{\overline{V}}{\sigma_v}, \qquad \text{(II-42)}$$
$$\beta_w = 0.6\frac{\overline{V}}{\sigma_w},$$

where β is restricted to $\beta \le 10$.

The horizontal components for peak wavelength are given by

$$\lambda_{m_u} = \lambda_{m_v} = \begin{cases} 1.5\, z_i\,, & \frac{z}{L} \le 0\,, \\ 0.7\, z_i\sqrt{\frac{z}{z_i}}\,, & \frac{z}{L} > 0\,. \end{cases} \qquad \text{(II-43)}$$

The constant, A, varies as a function of stability in the unstable boundary layer, and is computed from

$$A = \begin{cases} 0.31\,(1 - 3\frac{z}{L})^{-\frac{1}{3}}\,(1 - 15\frac{z}{L})^{0.25}\,(0.55 + 0.38\frac{z}{L})\,, & |\frac{z}{L}| \le 1\,, \\ 0.05\,(1 - 3\frac{z}{L})^{-\frac{1}{3}}\,(1 - 15\frac{z}{L})^{0.25}\,, & 0.1|\frac{z_i}{L}| > |\frac{z}{L}| > 1\,. \end{cases}$$
(II-44)

Values of $u'(t-\Delta t)$ and $v'(t-\Delta t)$ are determined randomly from a normal probability distribution with zero mean. However, the turbulence distribution of vertical velocity is skewed [Figure 10], and the normal distribution is modified by changing the method of computing w:

$$w(t - \delta t) = \begin{cases} \alpha\omega^+ + \dfrac{\omega^-}{\alpha} - \eta , & \dfrac{z}{L} \le 0 , \\ \dfrac{\omega^+}{\alpha} + \alpha\omega^- + \eta , & \dfrac{z}{L} > 0 , \end{cases} \qquad \text{(II-45)}$$

where

$$\begin{aligned} \alpha &= -0.028 + 0.6\,|P| , \\ \eta &= 0.54\,|P| , \end{aligned} \qquad \text{(II-46)}$$

and

$$P = \begin{cases} 0.1 + \dfrac{0.6}{0.68(1 - 15\frac{z}{L})^{-0.25} - 1.8\frac{z}{L}} , & \dfrac{z}{L} \le 0 , \\ 0.1 - 0.2(\frac{z}{L})^{0.2} , & \dfrac{z}{L} > 0 . \end{cases}$$

(II-47)

The variables ω^+ and ω^- are random values obtained from a normal probability distribution with zero mean and standard deviation, σ_w. As noted by Pielke (1984, p. 179), this method is not entirely satisfactory, since it depends on the particular random number generator routine.

3. The NPS Dispersion Model

As mentioned above, L, the Monin-Obukhov length was proscribed for this study, while V, the mean windspeed at a height of two meters was set to 3m/sec and the surface vegetative canopy roughness length was assumed to be 0.05m. Other flow parameters required by the McNider and NPS particle models were computed from recently developed boundary layer parameterizations, given here and by Panofsky and Dutton

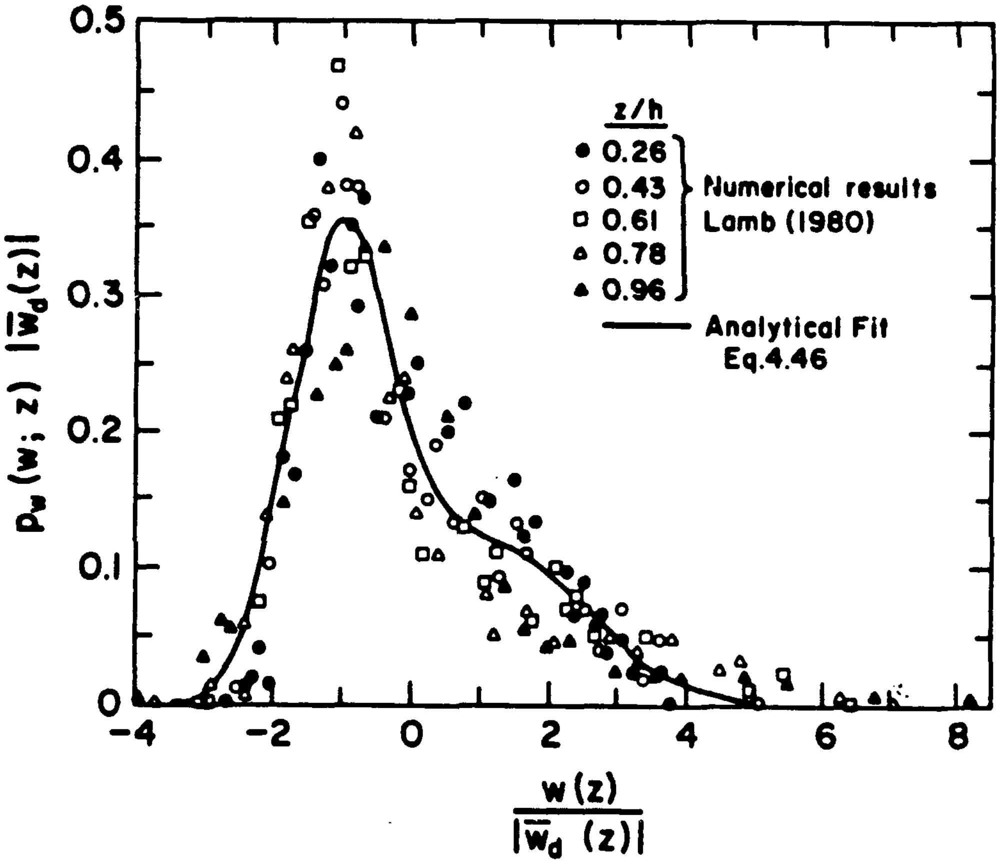

Figure 10. Probability density function of vertical velocity in the convective boundary layer. From Weil, Dispersion in the Convective Boundary Layer, Lectures on Air Pollution Modelling, Venkatram and Wyngaard, 1988.

(1984), Sorbjan (1990) and Kamada (1992a). More detailed descriptions of the underlying theory will be published in a forthcoming article. Many of the algorithms listed here are actually a part of the mesoscale windflow simulation which is also used to drive the McNider model. From this windflow simulation the McNider model obtains the following: the turbulent diffusivity, K_m, gradient Richardson number, Ri, vertical windshear, the windspeed at height z, v_z, and surface layer friction velocity, u_*.

u_* is computed from L, using the intermediate Businger function, ψ_m, as determined by equation II-24, and the square root of the surface drag coefficient,

$$C_{dn}^{1/2} = \frac{k}{\ln\left(\frac{z}{z_0}\right) - \Psi_m} \quad , \qquad \text{(II-48)}$$

so that

$$u_* = MAX(C_{dn}^{1/2}, 0.01) \quad . \qquad \text{(II-49)}$$

Given L and u_*, one can compute the surface vertical temperature flux,

$$\overline{w'\theta_0'} = -\frac{u_*^3\theta}{kgL} \quad . \qquad \text{(II-50)}$$

This allows an estimate of the free convective boundary layer turbulent velocity scale (for L < 0),

$$w_* \equiv \left(g z_i \frac{\overline{w'\theta'_0}}{\theta} \right)^{1/3} . \qquad \text{(II-51)}$$

This can be modified to include shear induced surface layer turbulence (forced convection) according to

$$w_{*s} = \left\{ \frac{0.4 - \frac{L}{z_i}\left[\frac{5}{4} - \ln\left|\left(1 - 150\frac{z_0}{L^{1/3}}\right) - 1\right|\right]}{0.4} \right\} \qquad \text{(II-52)}$$

(Kamada, 1992a).

For z/L < -0.5, i.e., above the surface layer, the following forms were used. The vertical velocity variance was given by

$$\sigma_w^2 = w_{*s}^2 \left[1.1\left(\frac{z}{z_i}\right)^{2/3}\left(1 - \frac{z}{z_i}\right)^{2/3} + R^{2/3}\left(\frac{z}{z_i}\right)^{2/3}\left(1 - \frac{z}{z_i} + D\right)^{1/3} \right] , \qquad \text{(II-53)}$$

where $R \simeq 0.2$ and $D \simeq 0.1$ are ratios of the inversion/surface temperature flux and entrainment zone/boundary layer depth (Sorbjan, 1990).

In a standard atmospheric windflow model .with second order turbulence closure scheme, the turbulence kinetic energy (tke) is computed prognostically. If σ_w^2 is supplied as above, the horizontal variances follow by subtraction. Without such a model, σ_u^2 and σ_v^2 are obtained diagnostically as follows.

The Andre (1978) third order turbulence closure used the following prognostic form for vertical velocity variance,

$$\partial\frac{\overline{w^2}}{\partial t} = -\partial\frac{\overline{w^3}}{\partial z} + 2\beta\overline{w'\theta'} - C_4\epsilon\left[\frac{\sigma_w^2}{\bar{e}} - \frac{2}{3}\right] - \frac{2}{3}\epsilon \quad . \qquad \text{(II-54)}$$

Assuming steady state and neglecting diffusion, this can be truncated to estimate the anisotropy, $\sigma_w^2/\bar{e}$ where $\bar{e}$ is the tke. Mason and Thompson's (1987) large eddy simulation of neutrally stable flow showed that $\sigma_w^2/\bar{e} \simeq 1/3$ near the surface and increased with height. This occurs because the surface restricts vertical motion. Also, most boundary layer turbulence results from "surface no-slip" induced shear which creates mostly horizontal tke. The increase with height occurs because the shear drops rapidly, so $\sigma_{w2}/\bar{e}$ gradually approaches the isotropic 2/3 value through pressure redistribution, consistent with Mason and Thompson's results.

Like the tke on which it depends, ϵ, the molecular dissipation rate of tke also decays gradually with height. For convective boundary layers, Sorbjan (1990) parameterizes ϵ as,

$$\phi\epsilon = \frac{3(1 - z/z_i)}{4} + Rz/z_i \quad , \quad \text{and} \qquad \text{(II-55)}$$

$$\epsilon = \phi\epsilon w_{*s}^3/z_i \quad . \qquad \text{(II-56)}$$

Putting the above together, the tke anisotropy ratio can be parameterized as,

$$\frac{\sigma_w^2}{\bar{e}} = \frac{2\epsilon_0 - \epsilon}{3\epsilon_0} + \frac{\beta \overline{w'\theta'}_z}{\epsilon} , \qquad \text{(II-57)}$$

(Kamada, unpublished). The first term on the right hand side accounts for the height dependence under neutral stability, while buoyancy flux in the second term accounts for non-neutral stability. This form actually corresponds to the Lenschow et al. (1980) measurements better than the Andre model itself or derivatives thereof (Therry and Lacarrere, 1980).

So the horizontal turbulent velocity variances become

$$\sigma_u^2 = (5/9)\sigma_w^2(2e/\sigma_w^2 - 1) \quad , \quad \text{and} \qquad \text{(II-58a)}$$

$$\sigma_v^2 = 0.8\sigma_u^2 \quad . \qquad \text{(II-58b)}$$

This allows one to compute ē, the turbulence kinetic energy (tke) as,

$$\bar{e} = (\sigma_u^2 + \sigma_v^2 + \sigma_w^2)/2 \quad . \qquad \text{(II-59)}$$

The molecular tke dissipation length scale is parameterized as

$$l_\epsilon = 1/(1/z + 1.4\epsilon\bar{e}^{1/2}) \qquad \text{(II-60)}$$

from which the turbulent mixing length is estimated as,

$$l_k = MIN(3l_\epsilon \; \sigma_w^2/\bar{e}, \; z) \quad . \qquad \text{(II-61)}$$

This finally yields the required turbulent diffusivity,

$$K_m = 0.44 l_k \bar{e}^{1/2} \quad , \tag{II-62}$$

(Kamada, 1992b). The buoyancy gradient for the unstable boundary layer is given by

$$\beta \partial \Theta / \partial z = 0.6\Theta_* / z_i ((1 - z/z_i)^{2/3} / (z/z_i)^{1/3} - 2R^{2/3} (z/z_i)^{2/3} / (1 - z/z_i + d)^{2/3}) \quad , \tag{II-63}$$

(Sorbjan, 1990). The above formulations were applied to the convective (unstable) boundary layer (z/L < -0.5) above the surface layer.

For the unstable surface layer (L < 0, but z/L > -0.5),

$$\phi_m = (1 - 28z/L)^{-1/4} \quad , \tag{II-64}$$

$$K_m = k u_* z / \phi_m \quad , \tag{II-65}$$

$$\bar{e} = (5.6 K_m / z_i)^2 \quad , \tag{II-66}$$

$$Ri = z/L \quad , \tag{II-67}$$

$$\phi_1 = (12 - 0.5 z_i / L)^{1/3} \quad , \tag{II-68}$$

$$\phi_2 = \phi_1 \quad , \tag{II-69}$$

$$\phi_3 = (1 - 14z/L)^{1/4} \quad , \tag{II-70}$$

$$\phi_\epsilon = (1 + .75 * (-z/L)^{2/3})^{3/2} \quad , \tag{II-71}$$

and finally

$\epsilon_0 = u_*^3\phi_\epsilon/z_i$, where the subscript 0 refers (II-72)

to the surface value. (Sorbjan, 1990).

For the neutral to stable boundary layer and surface layer, where $L \geq 0$,

$\alpha = 2 - 10/L$, and (II-73)

$\beta = 3 - 20/L$, (II-74)

(Kamada, 1992b). So that,

$\phi_1 = 2.2(1 - z/z_i)^{\alpha/2}$, (II-75)

$\phi_2 = \phi_1$, and (II-76)

$\phi_3 = 1.6(1 - z/z_i)^{\alpha/2}$. (II-77)

The local friction velocity and Monin-Obukhov lengths were characterized in Sorbjan (1990) as

$u_l = u_*(1 - z/z_i)^{\alpha/2}$, and (II-78)

$L_l = L(1 - z/z_i)^{3\alpha/2-\beta}$. (II-79)

and the temperature flux at height, z, was given by

$\overline{w'\theta'}_z = \overline{w'\theta'}_0(1 - z/z_i)^{\beta}$. (II-80)

The Brunt-Vaisala frequency at height, z, was given by

$$BVF = 4.3u_*\Theta_*^2(1 + 3.7z/L)(1 - z/z_i)^{\beta+\alpha}/z \quad . \qquad \text{(II-81)}$$

The tke dissipation rate at z was parameterized as

$$\phi_\epsilon = 3.6(1 + 3.7z/L)(1 - z/z_i)^\beta/z \quad \text{, and} \qquad \text{(II-82)}$$

$$\epsilon = \phi_\epsilon u_*^3 \quad . \qquad \text{(II-83)}$$

The Richardson number was estimated as

$$Ri = \frac{z}{5z + L} \quad , \qquad \text{(II-84)}$$

and the momentum diffusivity was given roughly by,

$$K_m = \frac{ku_*z(1 - z/z_i)}{1 + 5\ z/L} \quad , \qquad \text{(II-85)}$$

(Sorbjan, 1990). Both the unstable surface layer and neutral to stable boundary layer cases used:

$$\sigma_u = u_*\phi_1 \quad , \qquad \text{(II-86a)}$$

$$\sigma_v = \sigma_u \quad , \qquad \text{(II-86b)}$$

$$\sigma_w = u_*\phi_3 \quad , \qquad \text{(II-86c)}$$

$$\psi_h = 2\ln[\ (1 + (1 - 14z/L)^{1/2})/2\ ] \quad \text{, and} \qquad \text{(II-87)}$$

$$\Theta_z = \Theta + \Theta_*(\ \ln(z/z_0) - \psi_h\ )/k \quad \text{, where} \qquad \text{(II-88)}$$

$$\Theta_* = \overline{w'\Theta'}_0/u_* \quad . \qquad \text{(II-89)}$$

The mean wind at height z and its shear were estimated to be

$$V_z = u_*(\ln(z/z_0) - \psi_m)/k \quad , \text{ and} \qquad \text{(II-90)}$$

$$\partial V_z/\partial z = u_*(1 + 4.7z/L)^{\alpha/2}(1 - z/z_i)^{\alpha/2}/kz \quad , \qquad \text{(II-91)}$$

(Panofsky and Dutton, 1984 and Sorbjan, 1989).

The following details only the significant items distinguishing the NPS and McNider models. The NPS particle model utilized the above formulations for the velocity variances and tke rather than those from McNider. For the NPS particle model, the Lagrangian vertical timescale was also estimated differently according to

$$T_{lw} = \lambda_{mw}/\sigma_w \qquad (L > 0) \quad , \text{ and} \qquad \text{(II-92)}$$

$$T_{lw} = 0.3z_i/w_{*s} \qquad (L < 0) \quad . \qquad \text{(II-93)}$$

Recognizing that the horizontal/vertical eddy aspect ratio decreases with increasing stability in the convective boundary layer, the horizontal Lagrangian integral timescales were estimated by

$$T_{lu} = (2 - 40/L)\ T_{lw} \quad , \text{ and} \qquad \text{(II-94)}$$

$$T_{lv} = T_{lu} \quad . \qquad \text{(II-95)}$$

The McNider algorithms for horizontal integral timescales were retained for stable cases.

Rather than use the McNider formulations, the vertical velocity skewness was simulated by converting every fifth updraft into a downdraft, while accelerating the updrafts and decelerating the downdrafts. So the updraft/downdraft probability ratio was set to 60%/40%, while the updrafts were 50% faster than the downdrafts, in tune with atmospheric measurements.

Non-stochastic buoyant forcing was also added to the NPS particle model for non-zero density gradients. For pure buoyancy driven motion, the force balance is just

$$\frac{\partial w}{\partial t} = - g \frac{\delta\theta}{\theta} \quad , \tag{II-96}$$

where w is vertical velocity, and $\delta\theta$ is the change in potential temperature due to a displacement, δz, away from the particle's neutral buoyancy height in a domain where $\partial\theta/\partial z \neq 0$. Here the neutral buoyancy height is taken to be the initial release height of the particle. Then,

$$\partial w = - \frac{g}{\theta} \, \delta\theta \, \partial t \quad . \tag{II-97}$$

If the particle's "memory" were perfect, then after a time, t, its buoyancy forced velocity would simply be

$$w = - \frac{g}{\theta} \int_0^t \delta\theta \, \partial t \quad . \tag{II-98}$$

However, to simulate dilution by ambient fluid, the particle must gradually forget its neutral buoyancy level. Its mnemonic e-folding time will be T_{lw}, the integral eddy coherence time scale already used to compute the stochastic component. So at each time step the particle's memory dims by the factor,

$$M = e^{\frac{-\delta t}{T_{lw}}} \quad . \tag{II-99}$$

In this case, after one time step,

$$w_1 = -(g/\Theta)\,\delta\Theta_1 \delta t \quad , \tag{II-100}$$

as before. But for subsequent time steps, the solutions are

$$w_2 = -(g/\Theta)\,\delta t(M\delta\Theta_1 + \delta\Theta_2) \quad , \tag{II-101}$$

$$w_3 = -(g/\Theta)\,\delta t(M^2\delta\Theta_1 + M\delta\Theta_2 + \delta\Theta_3) \quad , \tag{II-102}$$

or in general,

$$w_m = -(g/\Theta)\,\delta t \sum_{1}^{m} \delta\Theta_i \, M^{m+1-i} \quad . \tag{II-103}$$

This series grows quickly with m. However, also observe that

$$w_{i+1} = Mw_i - (g/\Theta)\,\delta\Theta_{i+1}\delta t \quad . \tag{II-104}$$

So only the last velocity need be retained. The buoyant forcing then added to the standard stochastic component (adjusted for vertical velocity skewness) to obtain the total vertical velocity fluctuation.

4. Random Number Generator Kernel

The dispersion models utilize normally distributed random numbers to generate the fluctuating velocity. These fluctuations are scaled by the standard deviation of the velocity components to relate them to the turbulent kinetic energy of the flow field. The method used to generate random numbers in this study was a variation of the congruence method, and is outlined in program McNider, subroutines NORNG and STRNUM (Appendix C). To check the distribution of the random number generation routine, the count distribution was plotted versus standard deviation [Figure 11]. It appears to have a generally Gaussian shape with perhaps slight skewness to the left of center for the 3600 iterations used in this test.

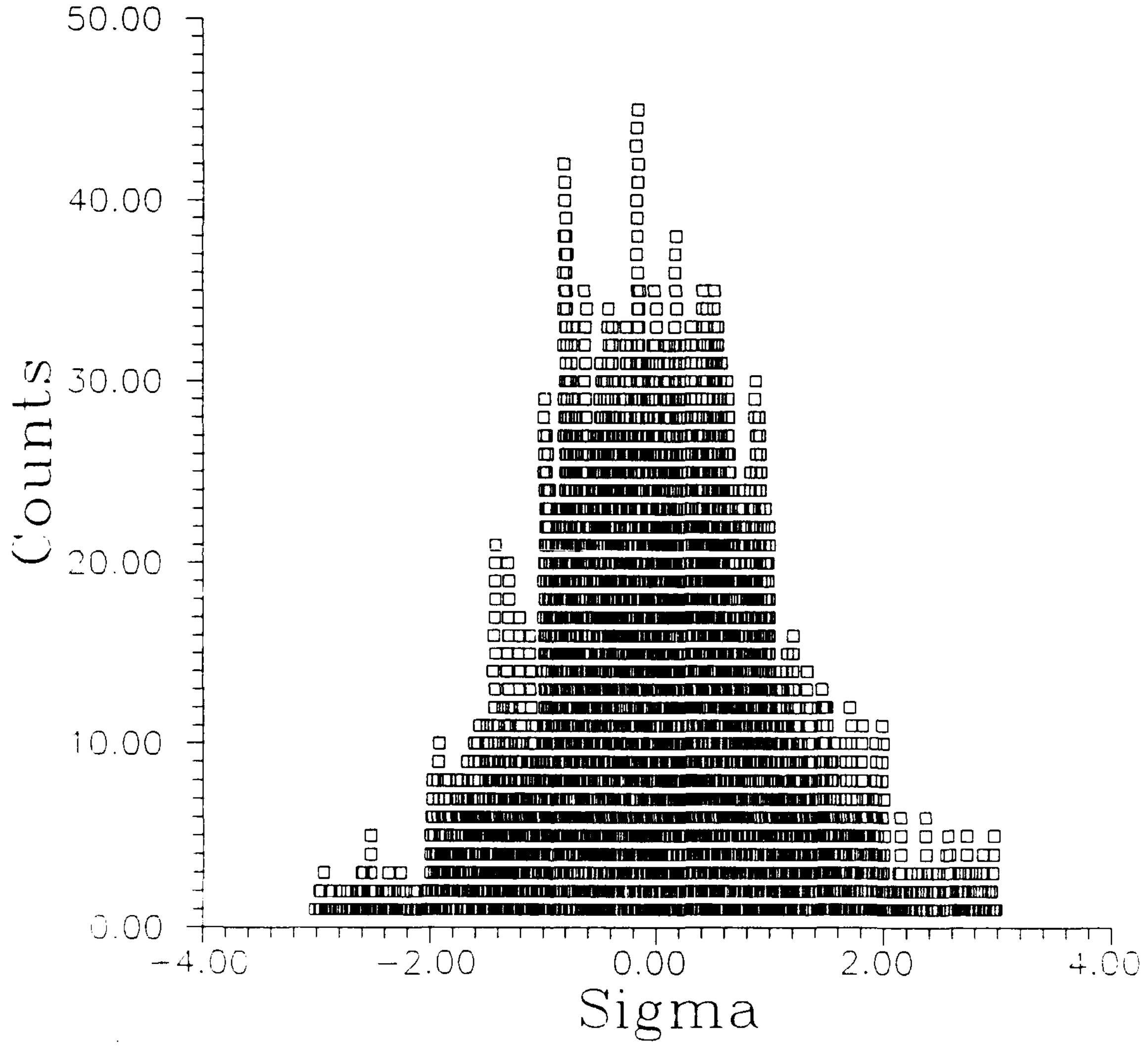

Figure 11. Random number distribution, Program McNider.

III. ANALYSIS

A. EQUIPMENT/SOFTWARE

The logistics difference equation, Henon equations, and the McNider Dispersion model were all written in standard FORTRAN-77, and run on Intel 386-based personal computers using the PROFORT compiler. All graphs were produced with Golden Software's Grapher and Surfer programs.

B. LOGISTICS DIFFERENCE AND HENON EQUATIONS

1. Methodology in analyzing the Logistics Difference Equation

Computer generated data from the logistics difference equation was produced using Program Feigenbm (Appendix 1). Since use of the logistics difference equation involves recursion, an iterative series rather than a time series is created, such that the fractal lengths were redefined using

$$L(i) = \frac{1}{\epsilon}\int_{0}^{T} |F(i+\epsilon) - F(i)|\, di \quad , \qquad \text{(III-1)}$$

which is approximated numerically by

$$L(i) = \sum_{i=k,k}^{N} |F(i+k) - F(i)|, \qquad \text{(III-2)}$$

where

$F(i)$ ≡ value of the function at the ith iterate,

i ≡ iteration step,

and k ≡ resolution in terms of number of iterations.

Analogous to the Lagrangian particle models studied later, the resulting iterative series represents a single particle, shifting position, with each successive iteration corresponding to a single time step. To insure independence from initial values, the program was run for 3700 iterations and the first 100 points were discarded. N was set to 3600 iterations for all plots, giving 3600 point data sets.

Since recursion does not permit non-integer iterations, the values of k had to be integer divisors of 3600. The 45 available values of k used are listed as an input data file in Program Feigenbm (Appendix 1).

From a series of k values for L(k) for a given μ, a standard linear regression routine then determined D_A. The standard deviation of D_A, was very small in chaotic situations, and large for during period doubling. The reasons for this are discussed below.

In computing D_A, the last three values of L(k) and k were discarded. On most plots, the curve flattened for $K > N/3$. This was also noted by DeCaria (1992) in analyzing time-series data. So in the parabolic logistics difference equation, one reason for discarding the highest k values is the rising number of fold-crossings with larger k. The "particle" trajectory is confined to the unit interval by the left-right symmetry or fold at mid-parabola. Therefore, the distance traversed during a fold crossing is not fully accounted as with a system without folds. The change in "apparent" traversed distance will be lessened as more fold crossings are involved. This results in decidedly non-linear slopes for log L(k) versus log k for k which are a sizeable fraction of the window.

2. Analysis

For values of μ corresponding to a fixed point, the plot of $\mathrm{Log}_{10}L(k)$ vs $\mathrm{Log}_{10}(k)$ is fairly flat, and the corresponding D_A value is small [Figure 12]. For values of μ corresponding to period doubling (period 2), the plot shows large jumps from a baseline, corresponding to multiple harmonics of 2 [Figure 13]. The baseline corresponds to a zero length. For these plots, the zero was adjusted by adding one to avoid a baseline at $-\infty$, a result of $\mathrm{Log}_{10}(0)$. This shift by one unit was important for later calculations; without it, $D_A \to \infty$ for all periodic functions.

A period 4 plot still shows regularity [Figure 14]. There appear to be three sets of overlapping slope patterns: one, a baseline of slope zero; the other two with similar slopes but different amplitudes. Again, this can be ascribed to harmonics. The length is zero for every fourth data point, and so will remain zero for harmonics having k values of 4, 8,

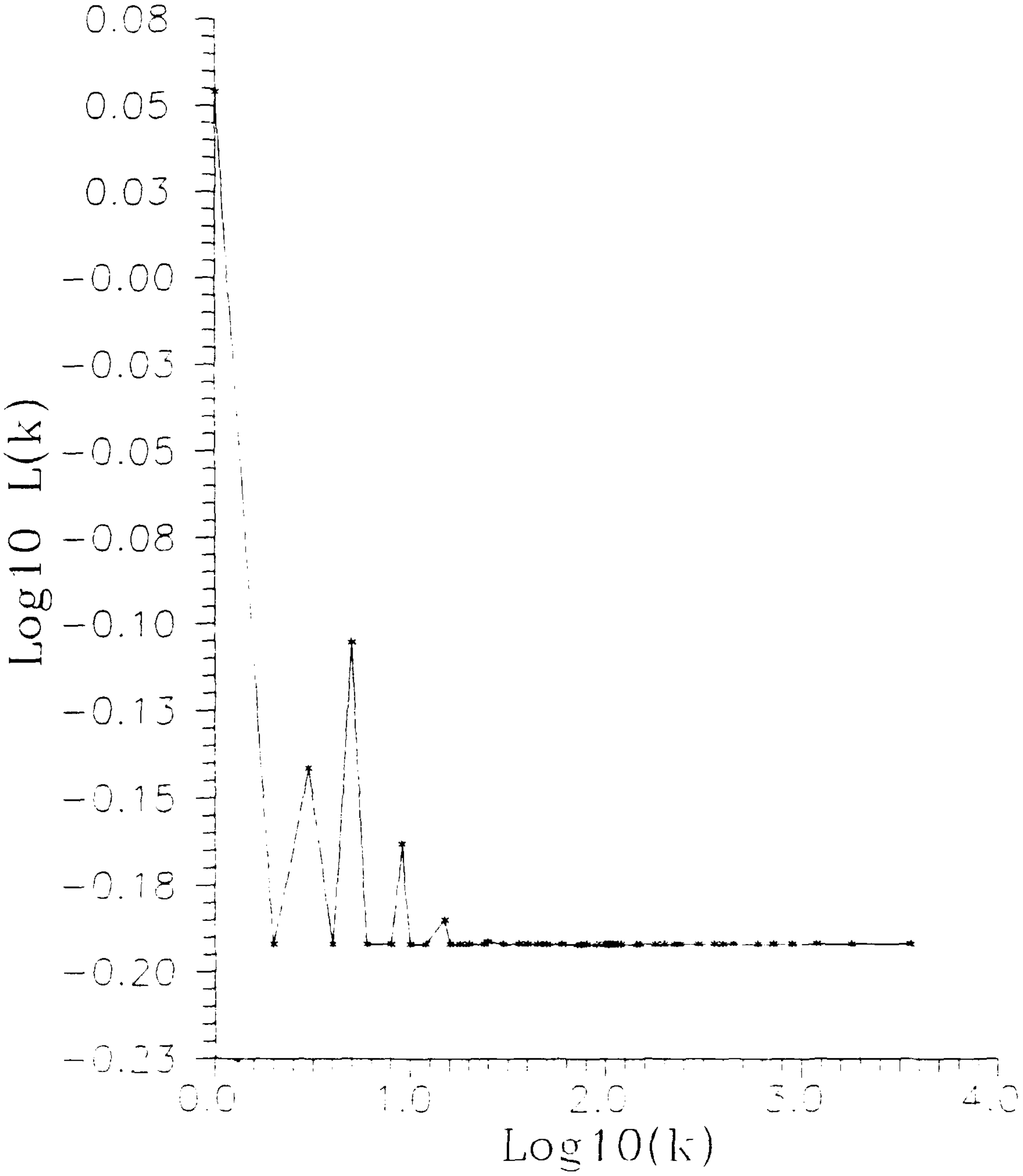

Figure 12. Logistics Difference Equation, $x_{n+1} = 2.8x_n(1-x_n)$, $x_0 = 0.01$, Fixed Point Attractor.

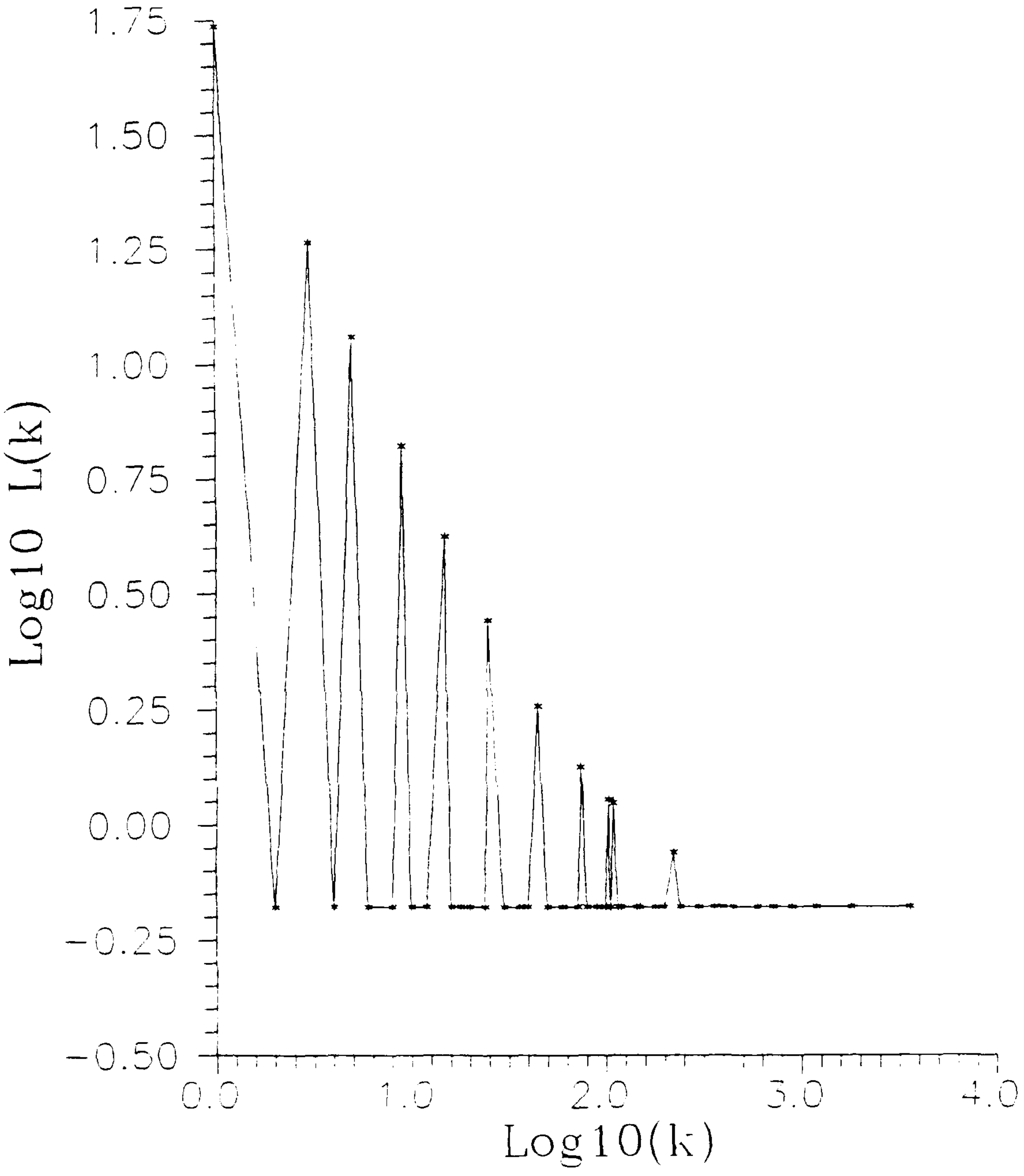

Figure 13. Logistics Difference Equation. $x_{n+1} = 3.0x_n(1-x_n)$, $x_0 = 0.01$, Period 2.

16, etc. The length is nonzero for every second data point, not including multiples of four. There is similarity in lengths for k values of 2, 6, 10, 18, etc. Likewise, there is similarity in lengths for k values which are not multiples of 2, e.g., 1, 3, 5, etc.

At the onset of chaos ($\mu = 3.569946$) as in Figure 15, all L(k) lift off the baseline; without periodicity, all lengths will remain non-zero. Thus, D_A will increase while σ_{D_a} decreases. Also noted is a sensitivity to initial conditions for identical values of μ, another characteristic of chaos. Figure 16, with $x_0 = 0.04$, is different from Figure 14 with $x_0 = 0.1$. However, Figure 15, with $x_0 = 0.1$, is virtually identical to Figure 17, with $x_0 = 0.01$. There may be some intrinsic pattern to this sensitivity to initial conditions. The fold crossing patterns might provide further insight. However, this question strays from the present focus.

For full chaos, σ_{D_a} becomes relatively small, and the plot becomes almost linear, as shown in Figure 18. This plot highlights why the last three points were discarded in computing D_A; the plot remains very linear sans large values of k. The main reason is the fold crossing patterns mentioned earlier. Additionally, not enough data points are retained in the length computation for large values of k to maintain similarity. For k=1200 and N=3600, the computed length is the sum of only three distance measurements, which is not enough

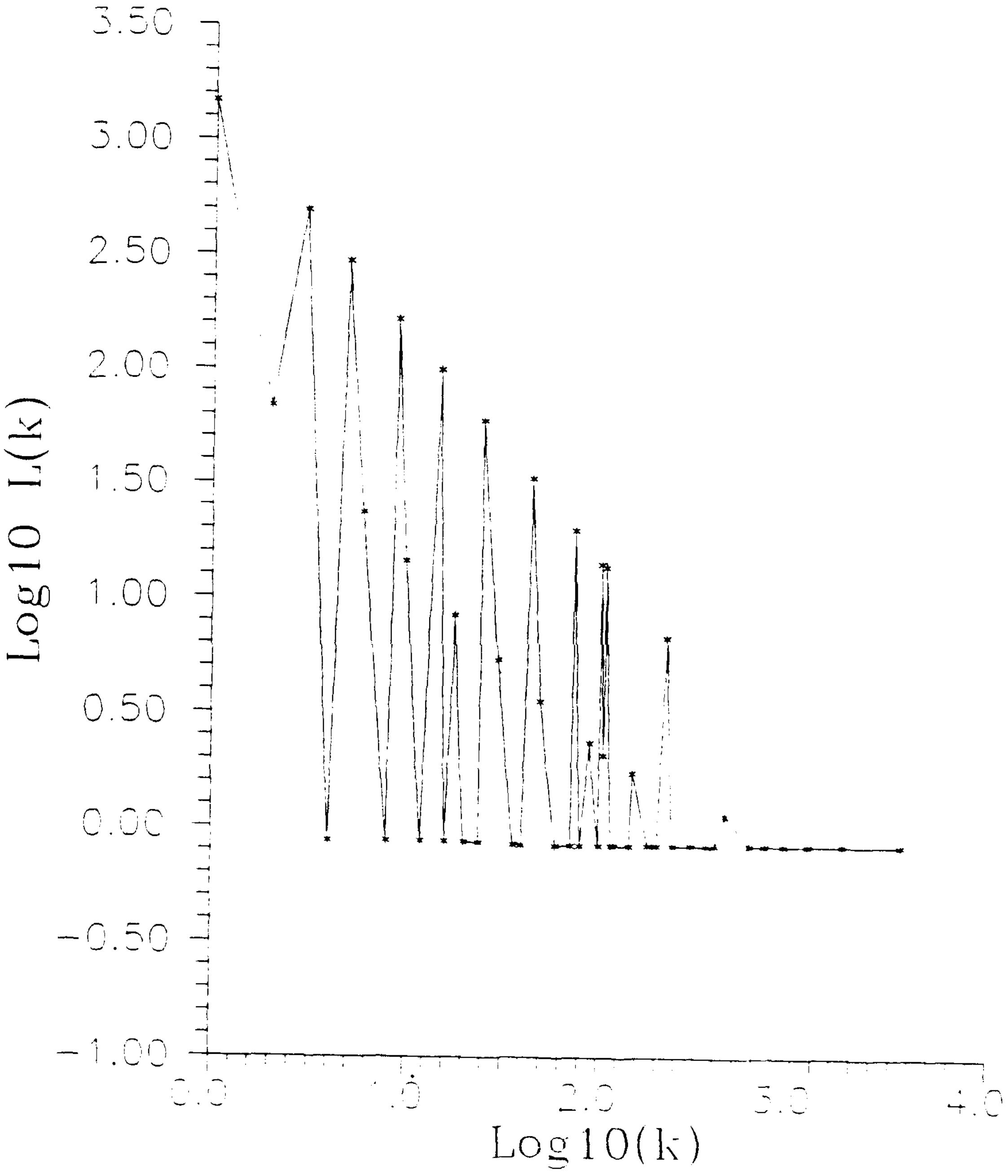

Figure 14. Logistics Difference Equation, $x_{n+1} = 3.48x_n(1-x_n)$, $x_0 = 0.01$, Period 4.

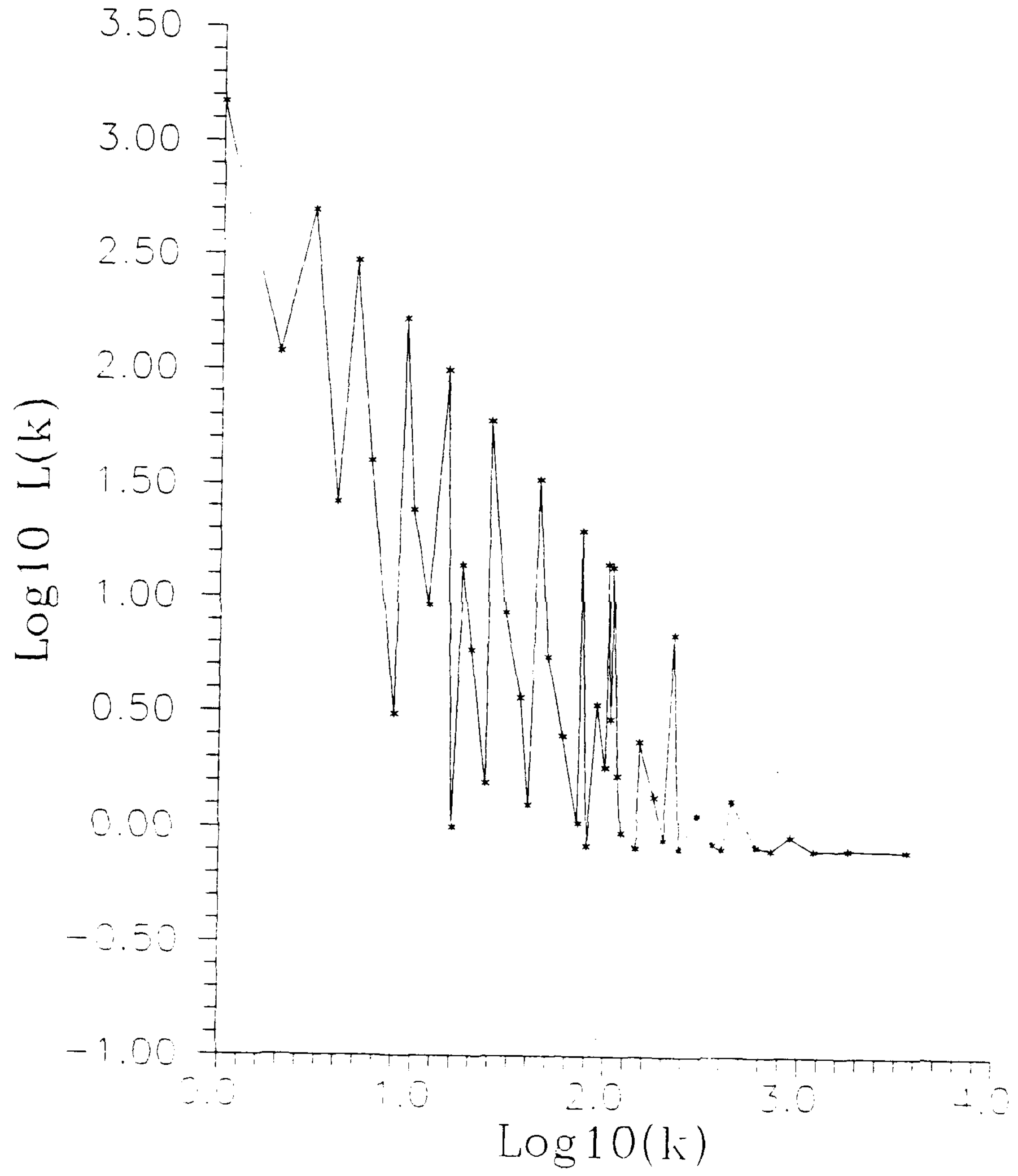

Figure 15. Logistics Difference Equation, $x_{n+1} = 3.56994568x_n(1-x_n)$, $x_0 = 0.1$, Onset of Chaos.

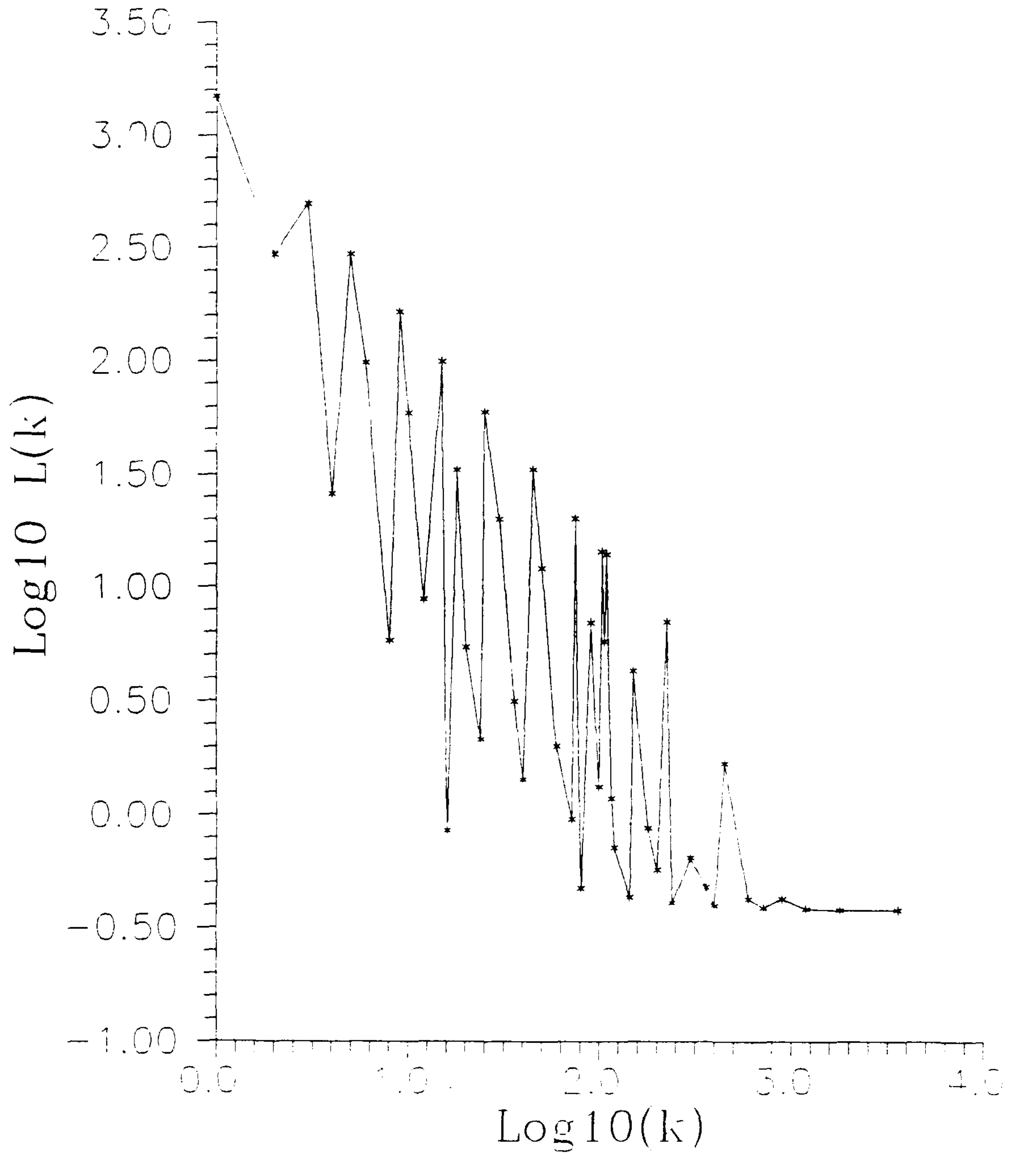

Figure 16. Logistics Difference Equation, $x_{n+1} = 3.56994568x_n(1-x_n)$, $x_0 = 0.04$, Onset of Chaos

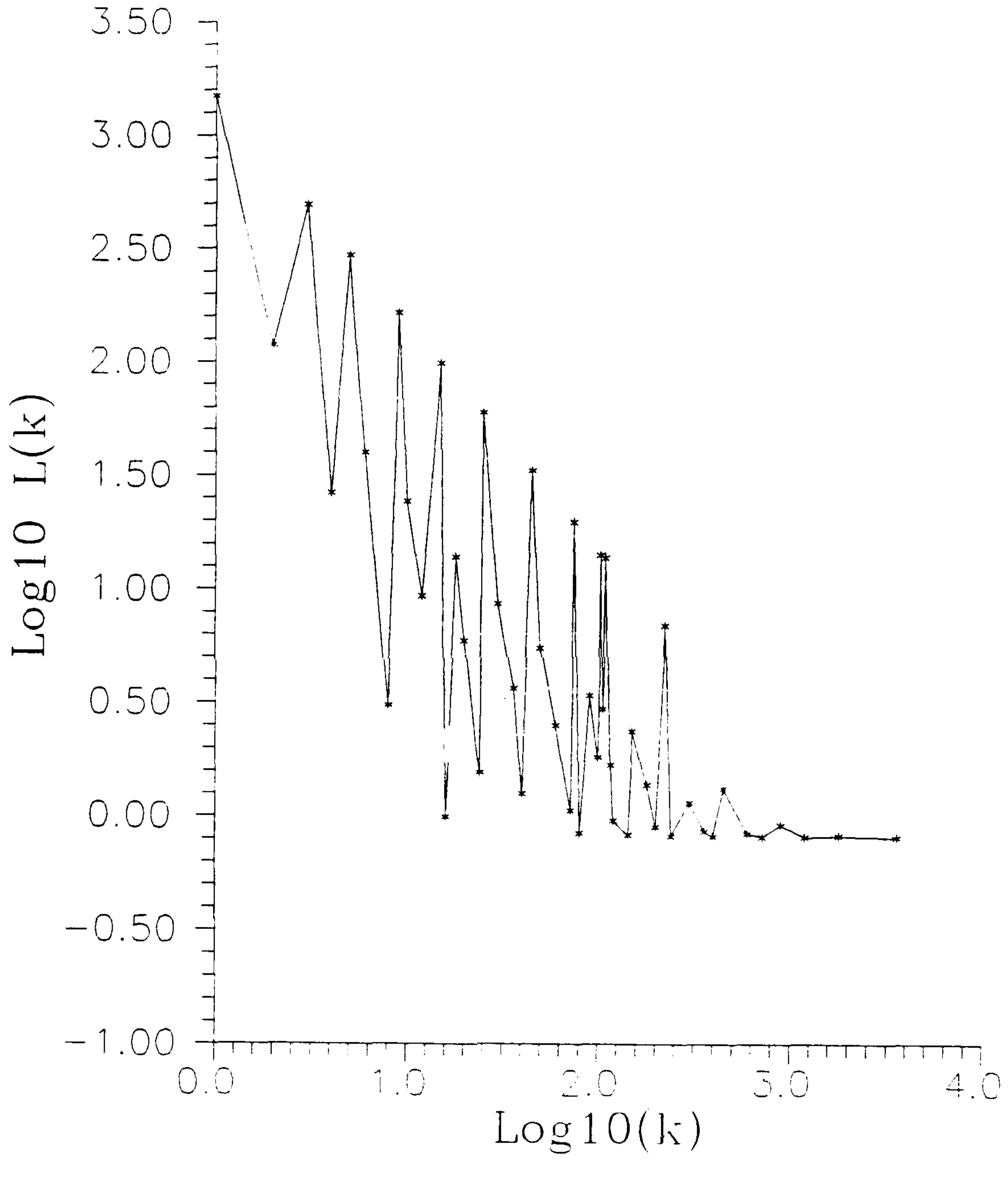

Figure 17. Logistics Difference Equation, $x_{n+1} = 3.56994568x_n(1-x_n)$, $x_0 = 0.01$, Onset of Chaos.

to develop a good statistical mean length.

A plot of D_A versus μ shows many of the features noted in the previous bifurcation plot [Figure 19]. At $\mu \approx 3.0$ D_A jumps from zero (corresponding to a fixed point) to ≈ 0.58, corresponding to period 2. At $\mu \approx 3.45$ is seen a transition to period 4. Higher μ values display regions of periods 8 and 16. However, at this level of resolution period 32 or higher order bifurcations cannot be discerned. Within the chaos regime, about $\mu \approx 3.6$, D_A seems to oscillate, and shows several sharp peaks and dips. The apparent window at $\mu \approx 3.85$ corresponds to a period 4 oscillation.

A plot of entropy versus μ shows similar features [Figure 20]. Regions of period 2, 4, and 8 are readily apparent. As in the D_A versus μ plot [Figure 19], differences between period 16 and above and chaos cannot be discerned at this level of resolution. The large window of low periodicity at $\mu \approx 3.85$ is also seen, as are several other windows of periodicity at roughly $\mu \approx 3.63$ and $\mu \approx 3.73$.

The Lyapunov exponent also shows the trends seen in the fractal dimension and entropy [Figure 21]. λ dips sharply at $\mu \approx 3.25$, 3.5, and 3.55, the centers for periods 2, 4, and 8. Again, changes beyond period 16 are hard to resolve. In the period doubling region, $\lambda = 0$ indicates that the function is transitioning to the next bifurcation, a fixed point fissioning to two fixed points. For $\lambda > 0$, the function is

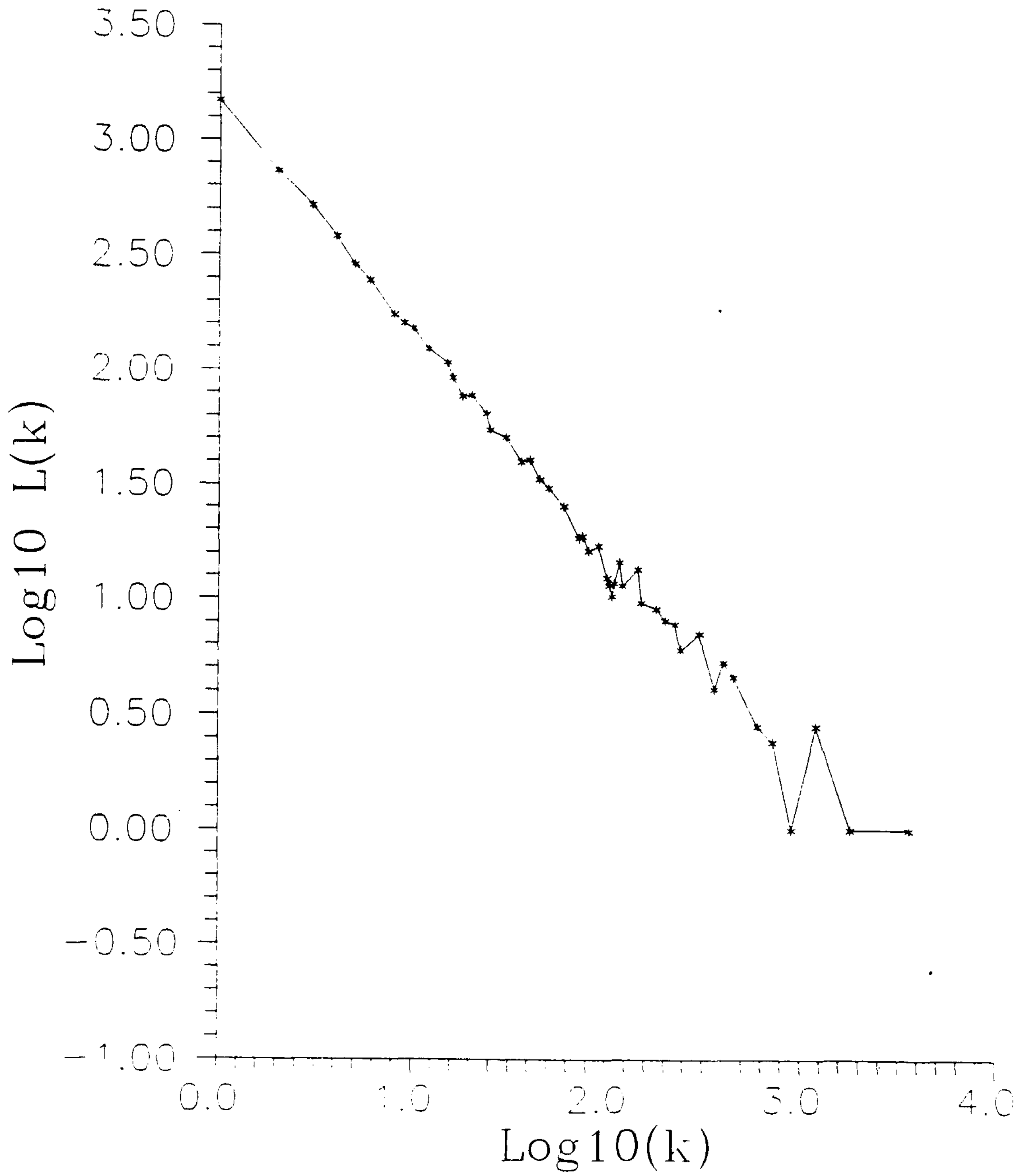

Figure 18. Logistics Difference Equation, $x_{n+1} = 3.9996x_n(1-x_n)$, $x_0 = 0.01$, Chaos.

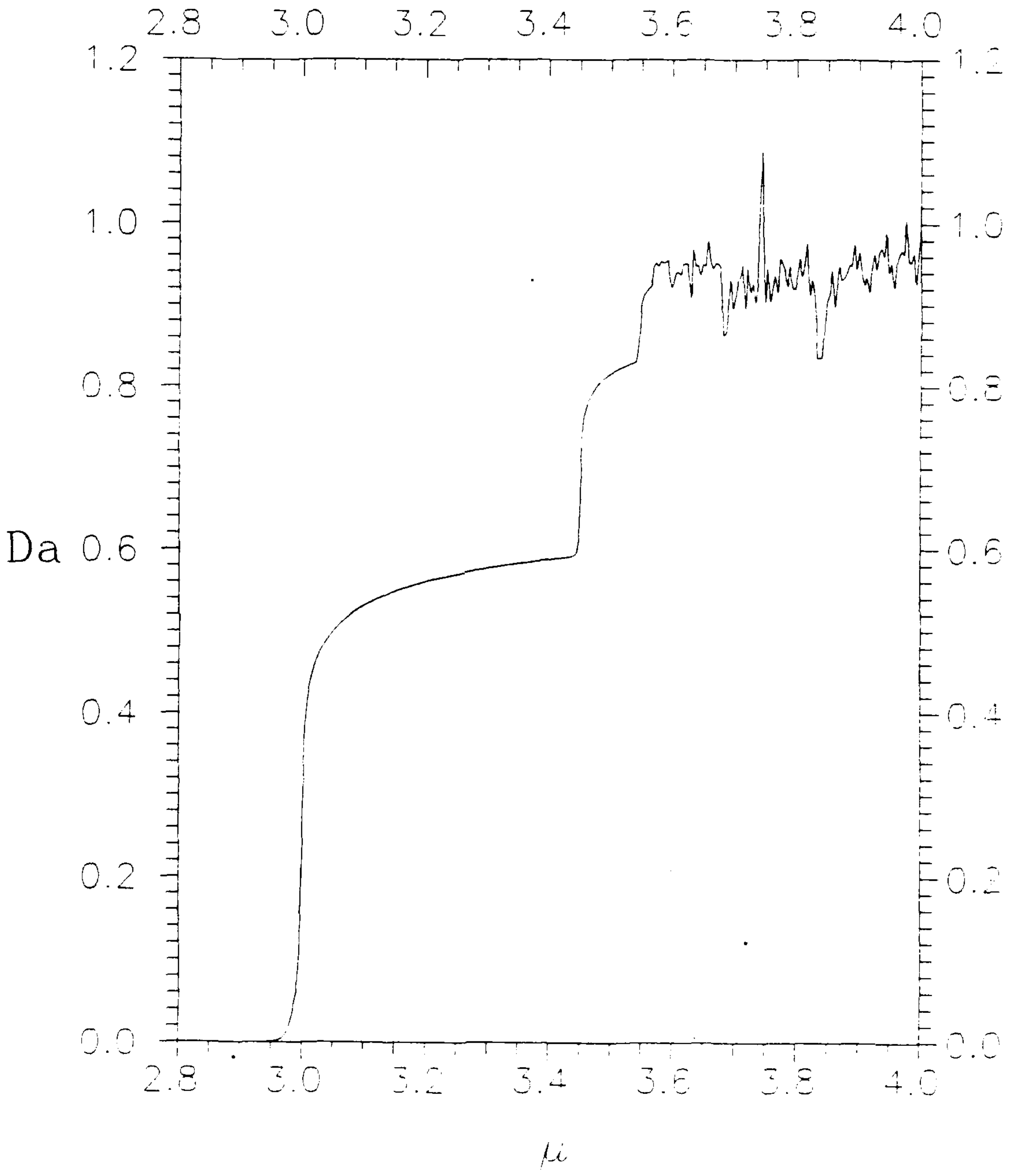

Figure 19. Fractal Dimension Analysis of Logistics Difference Equation, $x_{n+1} = \mu_n(1-x_n)$.

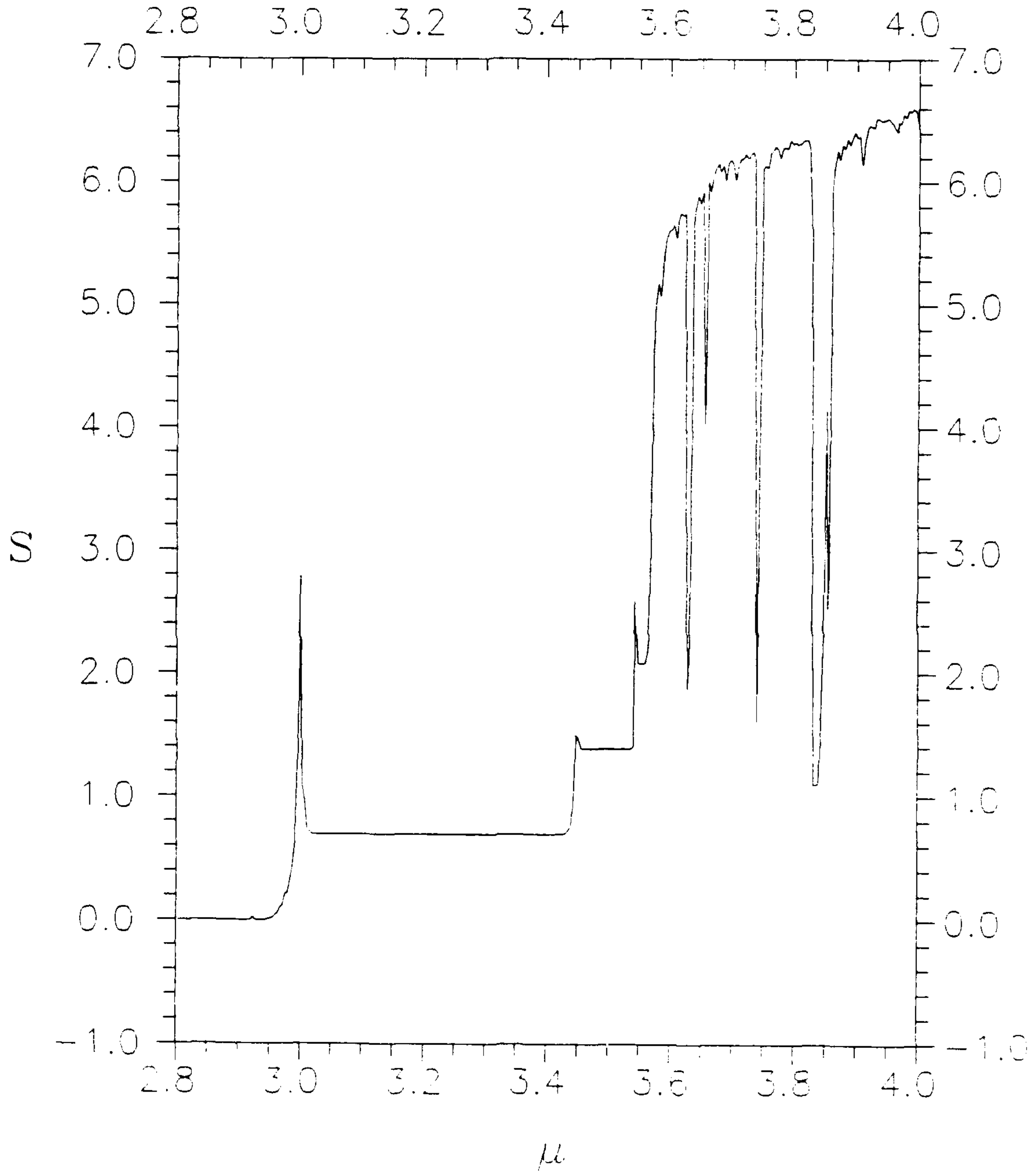

Figure 20. Shannon Entropy Analysis of Logistics Difference Equation, $x_{n+1} = \mu x_n(1-x_n)$.

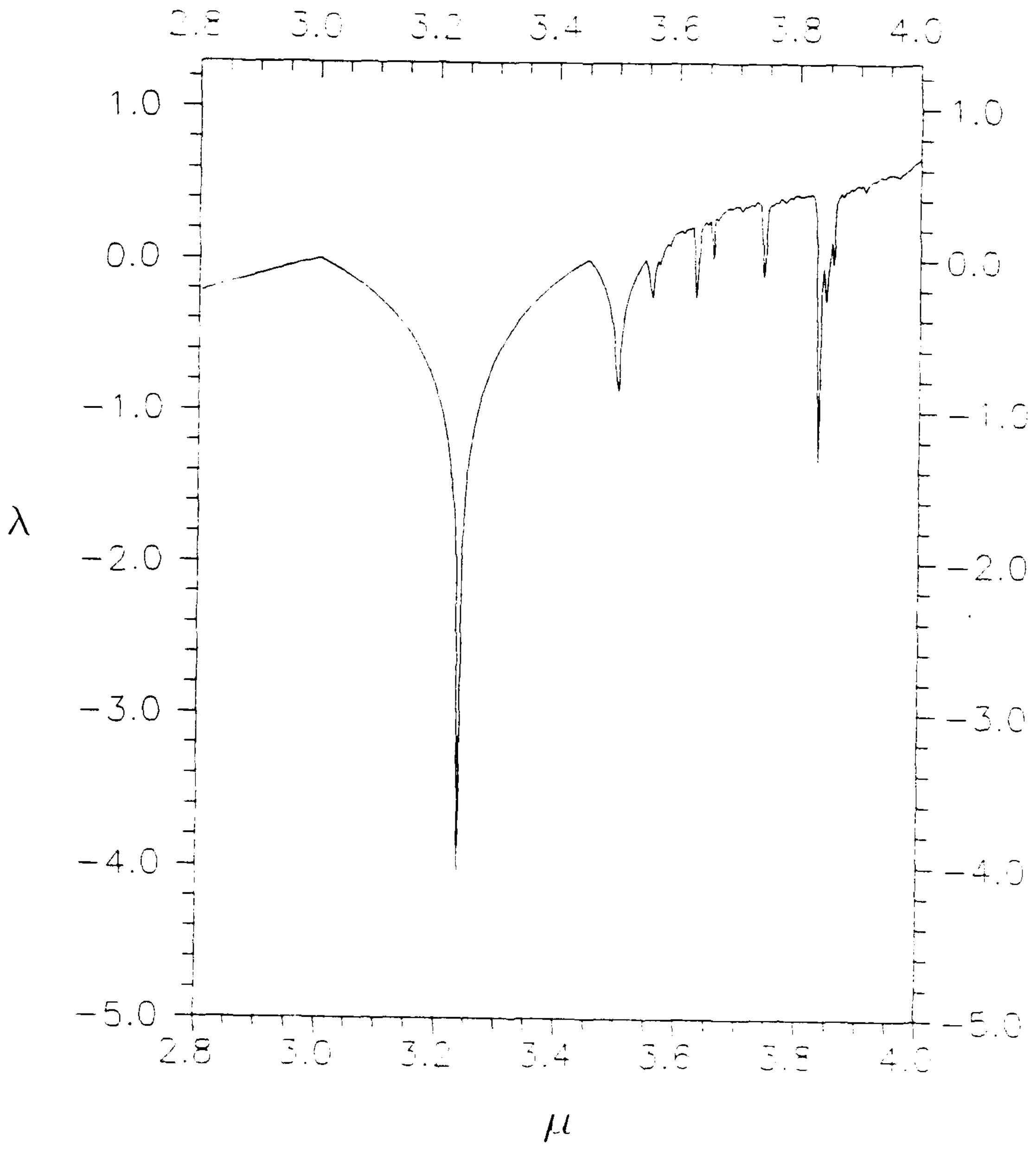

Figure 21. Lyapunov Exponent (λ) Analysis of Logistics Difference Equation, $x_{n+1} = \mu x_n(1-x_n)$.

chaotic. There are also numerous "windows of periodicity" in the chaotic region which appear in the bifurcation, fractal dimension, and entropy plots. These windows are more readily identified by the Lyapunov exponent; if λ drops below zero, the function is periodic.

Figure 22 shows the strong correspondences between entropy, fractal dimension, and Lyapunov exponent. Periodicities and chaos are apparent with each metric, as are some differences. D_A sometimes peaks where S and λ dip, as at $\mu \approx 3.725$. The low values of λ and S indicate periodic regions, for which D_A sometimes sharply increases. The bifurcation map [Figure 6] provides no clue as to the cause. However, if the fixed points are more widely separated than the average distance between successive points in the surrounding chaos regions, D_A will become relatively large. This is less likely for real fluids because motion is then constrained by physical forces and conservation laws. However, it points to a possible schism between diffusion and dispersion rates even without mean flow.

3. Henon Function Analysis

Since the Henon equations are a 2-D extension of the logistics difference equation, similar behavior is expected. With two dimensions, the length measurement is modified to

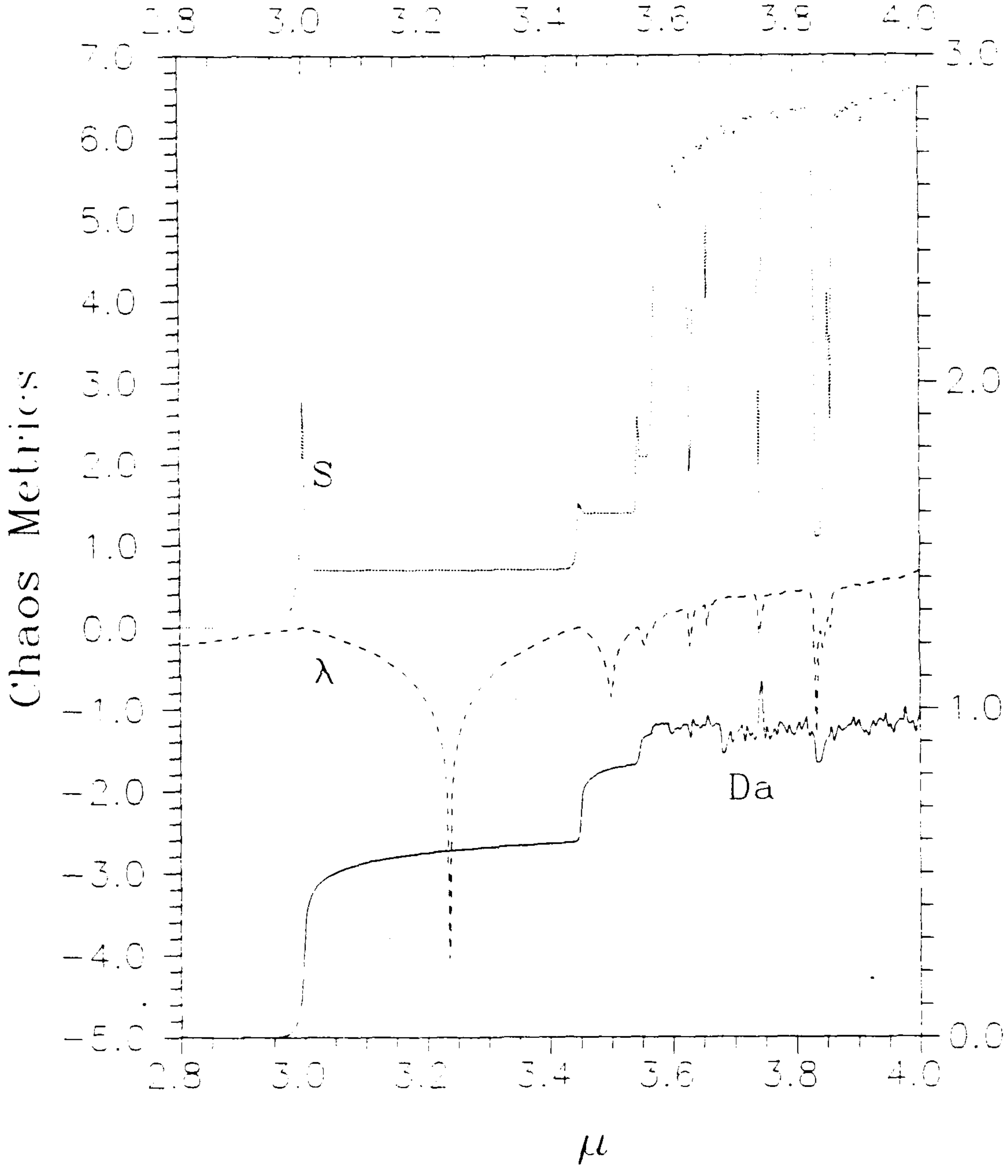

Figure 22. Fractal Dimension (D_A), Shannon Entropy (S), and Lyapunov Exponent (λ) for Logistics Difference Equation. Left axis λ; right axis S and D_A.

$$\Delta r = \sqrt{(x_{n+1}-x_n)^2 + (y_{n+1}-y_n)^2} \quad . \qquad \text{(III-3)}$$

In the period 2 case [Figure 7], a plot of $\mathrm{Log}_{10}L(k)$ vs $\mathrm{Log}_{10}k$ is very similar to the logistics difference equation for period 2 [Figure 23]. Again, for every k, evenly divisible by 2, the length drops to zero. There are two sets of slope patterns: a baseline with slope zero, and an upper non-zero slope (corresponding to non-even k-values). This upper slope appears to be quite straight, except for the largest k value.

At the onset of chaos [Figure 24], again the baseline lifts, and the overall deviation in slope is smaller [Figure 25]. It is also noted that $\mathrm{Log}_{10}L(k)$ only approaches zero when k is quite large. As noted before, for large k there are not enough distance measurements to adequately define a good statistical mean length; nor are there enough points to adequately define a periodicity.

At full chaos [Figure 7], the plot of $\mathrm{Log}_{10}L(k)$ vs $\mathrm{Log}_{10}k$ is so linear that, in regions of full turbulence or chaos, D_A can be defined with only two data points, *i* and *o*, corresponding to one small inner and one large outer scale of k. [Figure 26]. Then D_A in this case is given by

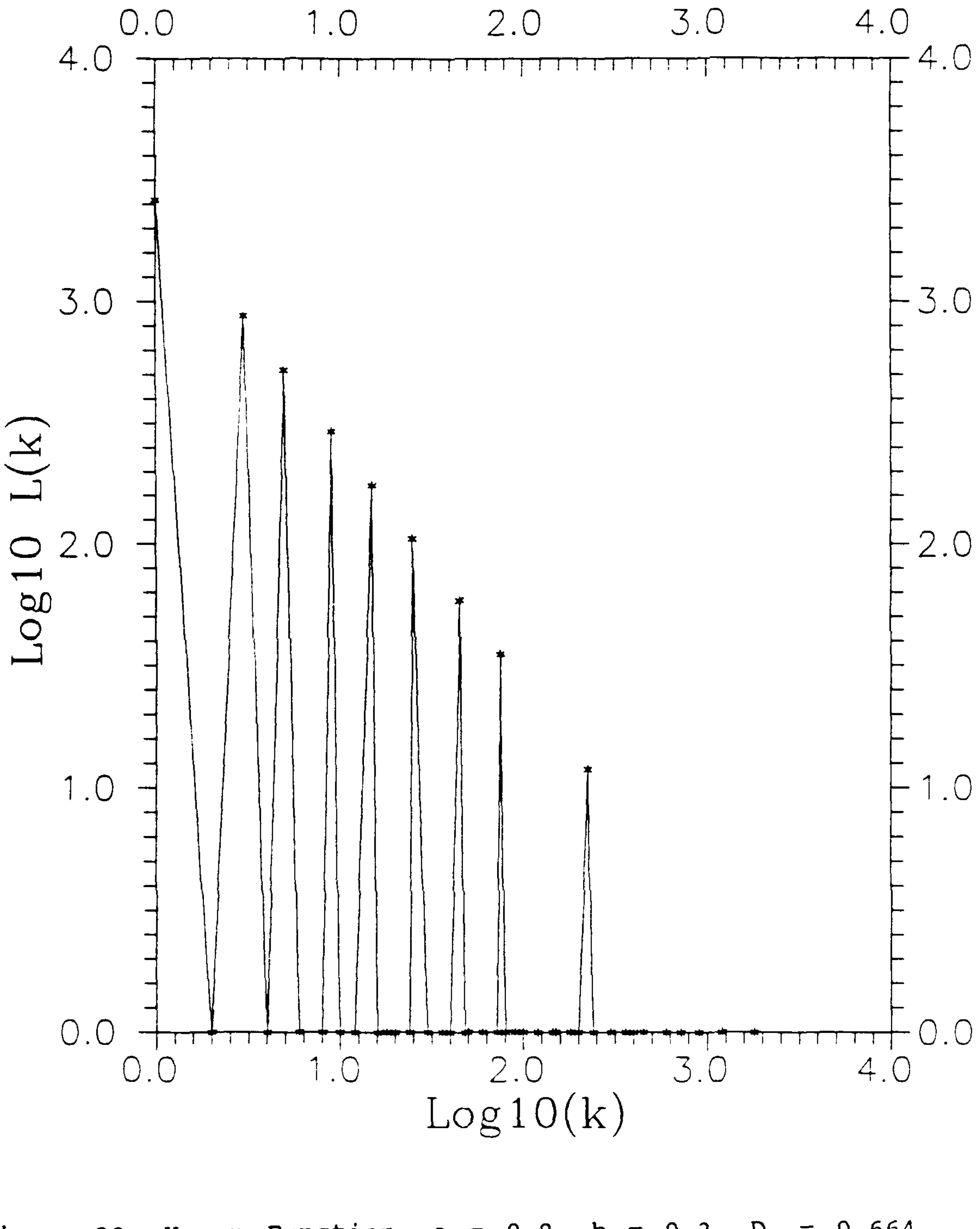

Figure 23. Henon Function, a = 0.8, b = 0.3, D_A = 0.664, Period Doubling.

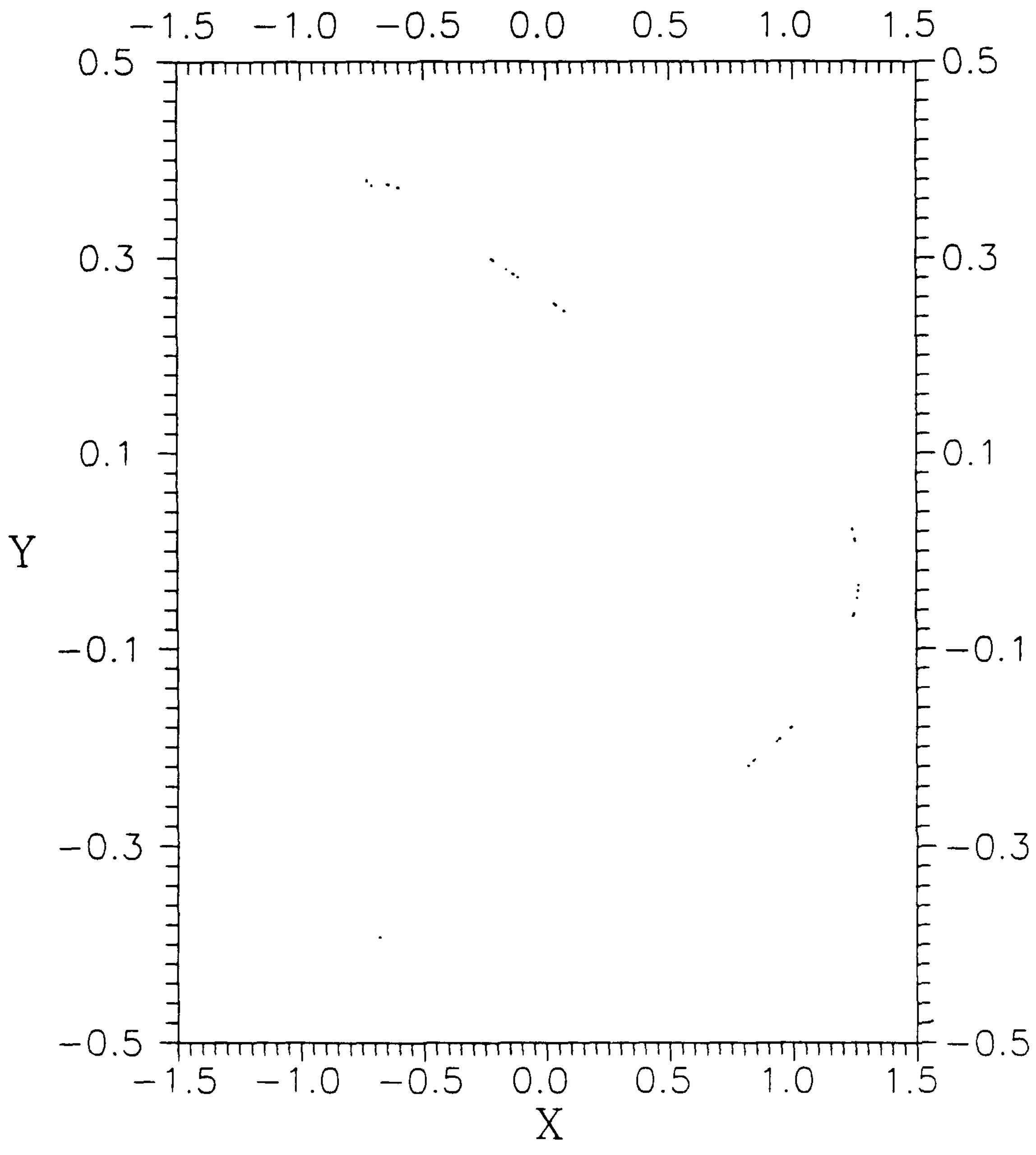

Figure 24. Henon Function, a = 1.057, b = 0.3, Onset of Chaos.

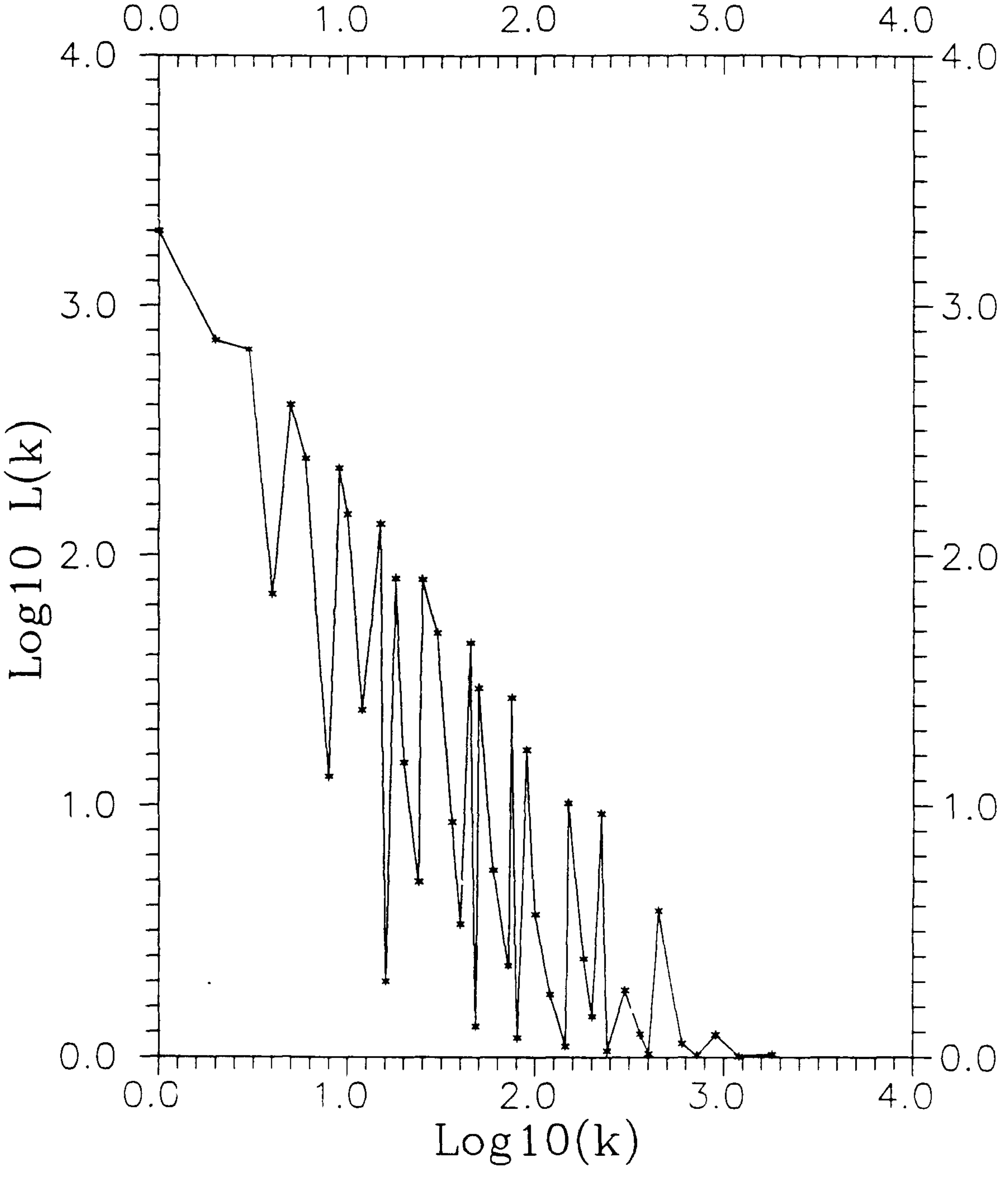

Figure 25. Henon Function, a = 1.057, b = 0.3, D_A = 1.114, Onset of Chaos.

$$D_A \equiv -\frac{\partial \ln\left[\frac{1}{\epsilon}\int_0^N |F(i+\epsilon) - F(i)|\,di\right]}{\partial \ln \epsilon}\ ,$$

$$\approx \frac{\ln\left(\frac{L_0}{\epsilon_0}\right) - \ln\left(\frac{L_i}{\epsilon_i}\right)}{\ln\epsilon_0 - \ln\epsilon_i} = \frac{\ln\left(\frac{L_i}{L_0}\right)}{\ln\left(\frac{\epsilon_0}{\epsilon_i}\right)} - 1\ . \qquad \text{(III-4)}$$

This shows immediately that an increase in D_A implies a relative increase in the apparent magnitude of the fluctuations discernible at higher resolutions, i.e., a shift in the amplitude spectrum toward smaller scales. This interpretation of D_A will be of use in analyzing the atmospheric Lagrangian particle models.

There appears to be a close correspondence between entropy S and D_A for the Henon function [Figure 27]. Both show jumps corresponding to periods 2, 4, 8, and perhaps even for period 16. Both plots show corresponding changes when there is periodicity within the chaos region.

A higher resolution look at the region, $1.052 \le a \le 1.082$, displays some interesting complexity [Figure 28]. Not only are there normal bifurcations, but at $a \approx 1.062$ a new branch appears with no obvious connection to previous points. This does not correspond to the fissioning process observed earlier, but rather to an entirely new set of solutions. At $a \approx 1.080$, points from this new branch break away and drift toward the original branch.

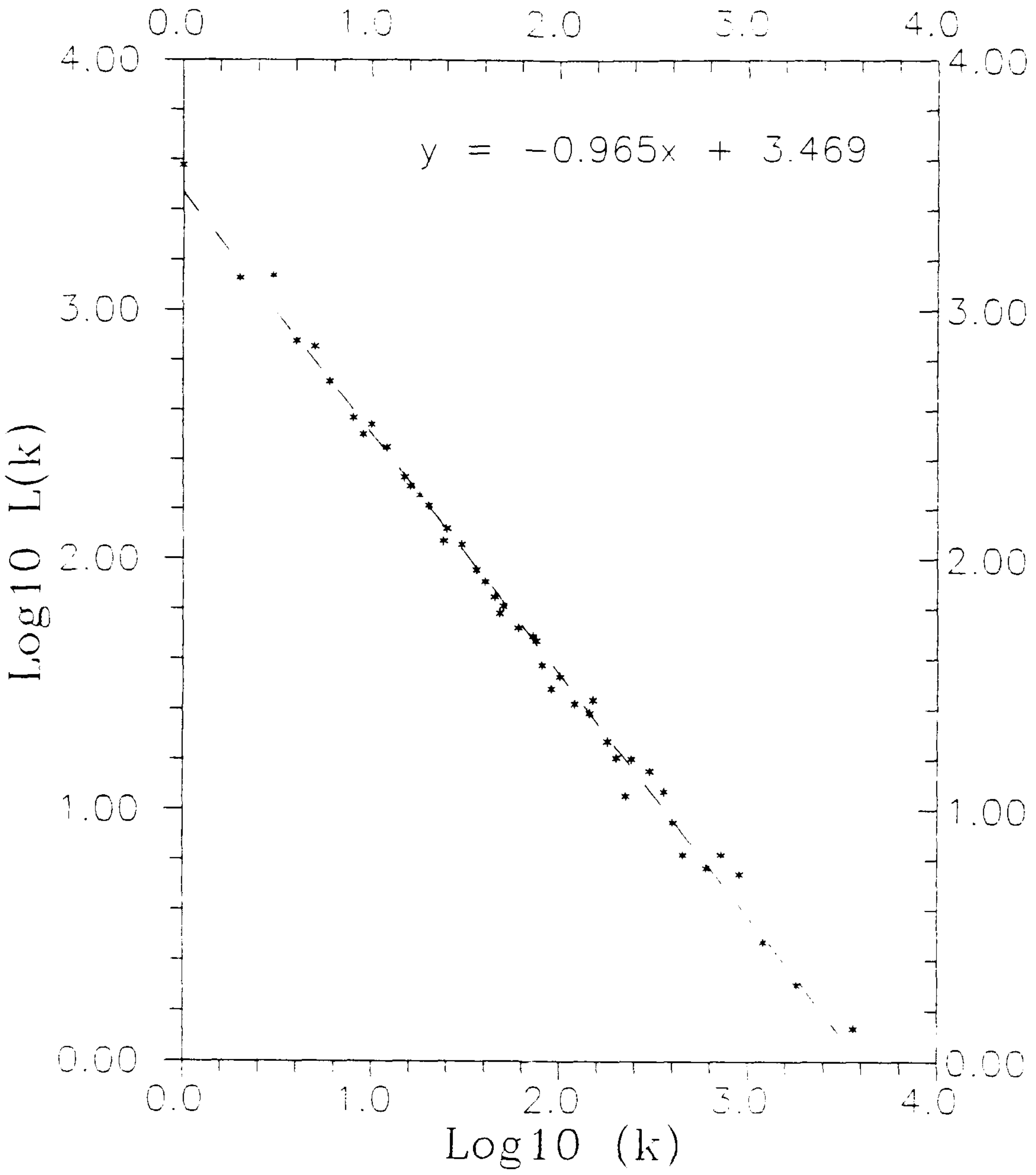

Figure 26. Henon Function, a=1.4, b=0.3, D_A=0.965.

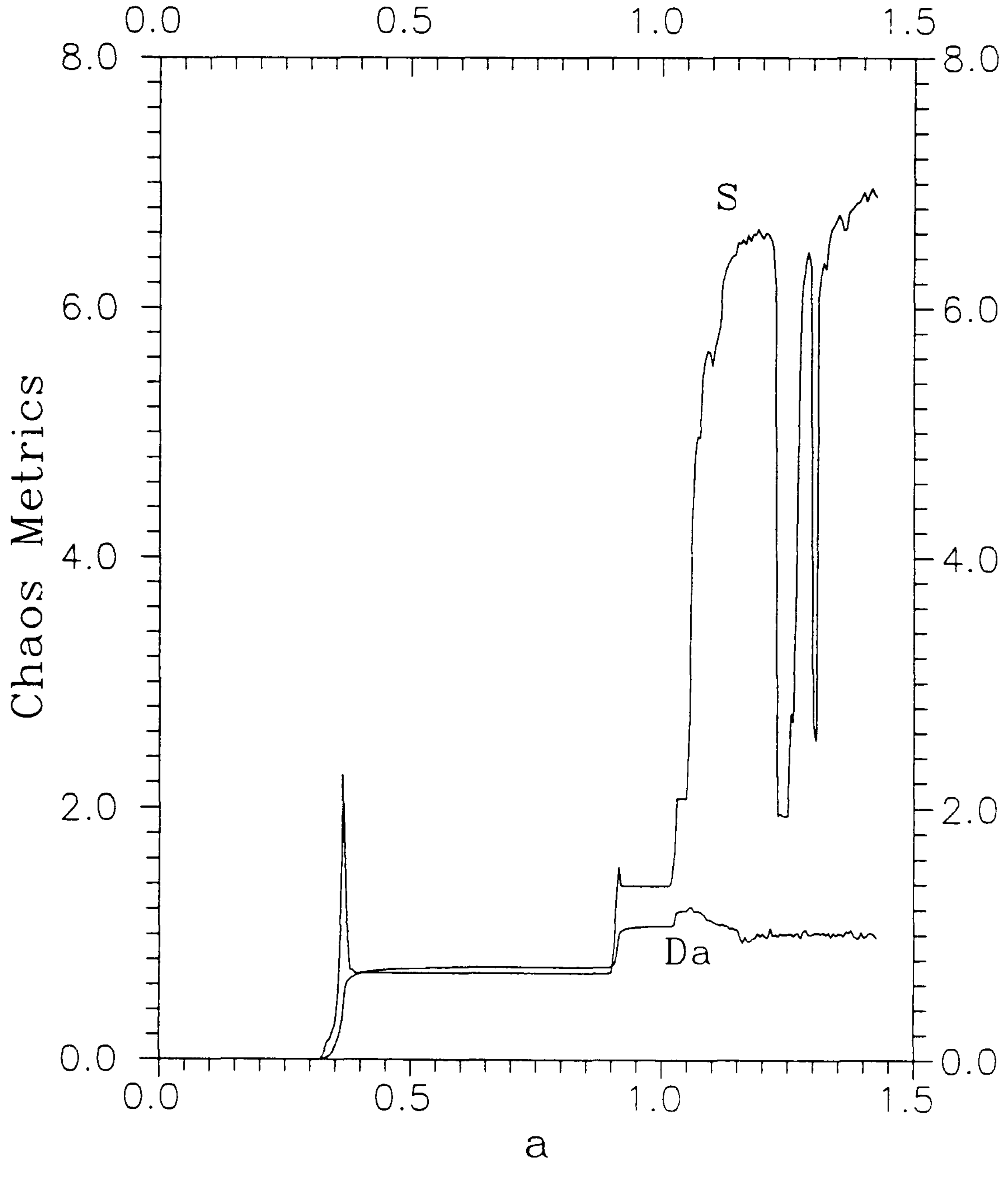

Figure 27. Comparison of Fractal Dimension (D_A), and Shannon Entropy (S) for the Henon Function, with b=0.3.

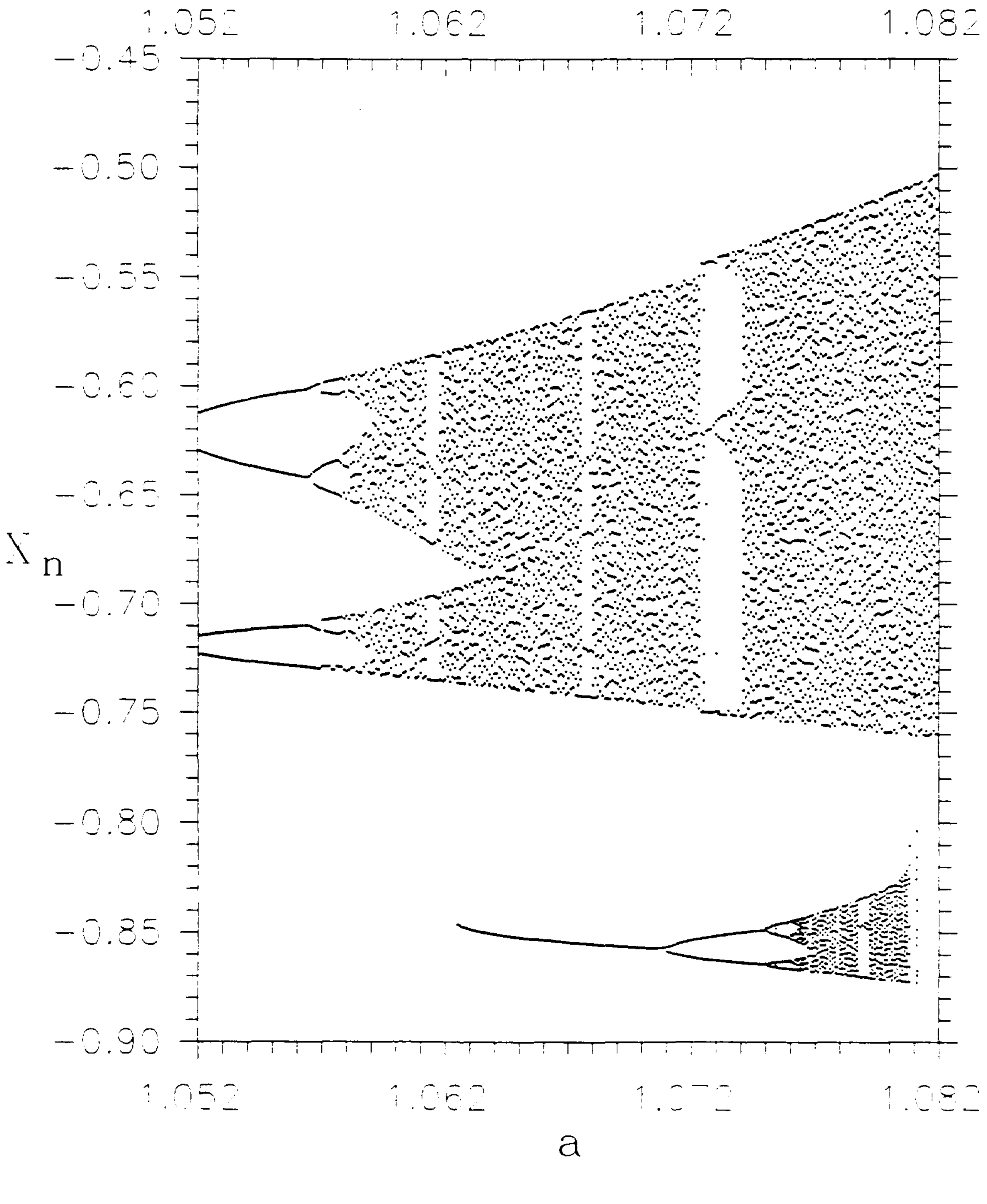

Figure 28. Henon Function Bifurcation Map, with b = 0.3.

In the region, $1.052 \leq a \leq 1.082$, D_A jumps in amplitude whenever the bifurcation map shows a window of periodicity [Figure 29]. D_A based on x rather than the total distance r portrays much but not all the same information: jumps correspond to most of the same windows of periodicity [Figure 30]. The strong peaks in D_A correspond to sharp dips in S; however, sharp dips in S do not necessarily correspond to sharp peaks in D_A [Figure 31]. So a jump in average distance traversed between iterations always accompanies a drop in position randomness but not vice versa. This suggests that periodicity in the Henon system sometimes results from the sudden appearance of fixed points which are as closely spaced or more closely spaced than the mean distance between successive iterations in the surrounding chaos.

Recognizing that there are two free parameters in the Henon equation (a and b), D_A and S were mapped for values of $0 \leq a \leq 1.5$ and $0 \leq b \leq 1.0$ [Figures 32, 33]. The two 3-d maps are remarkably similar, and suggest that $\exp(D_A)$ would be of the same order as S. The regimes of low periodicity are well defined. Where D_A and S are zero, the Henon function has period 1, corresponding to one fixed attractor. The first jumps to periods 2 and 4 are well defined at this level of resolution. There are other steps of discernible width for high a and b values, on the left and back sides of the graphical "mountain". Another interesting feature is the

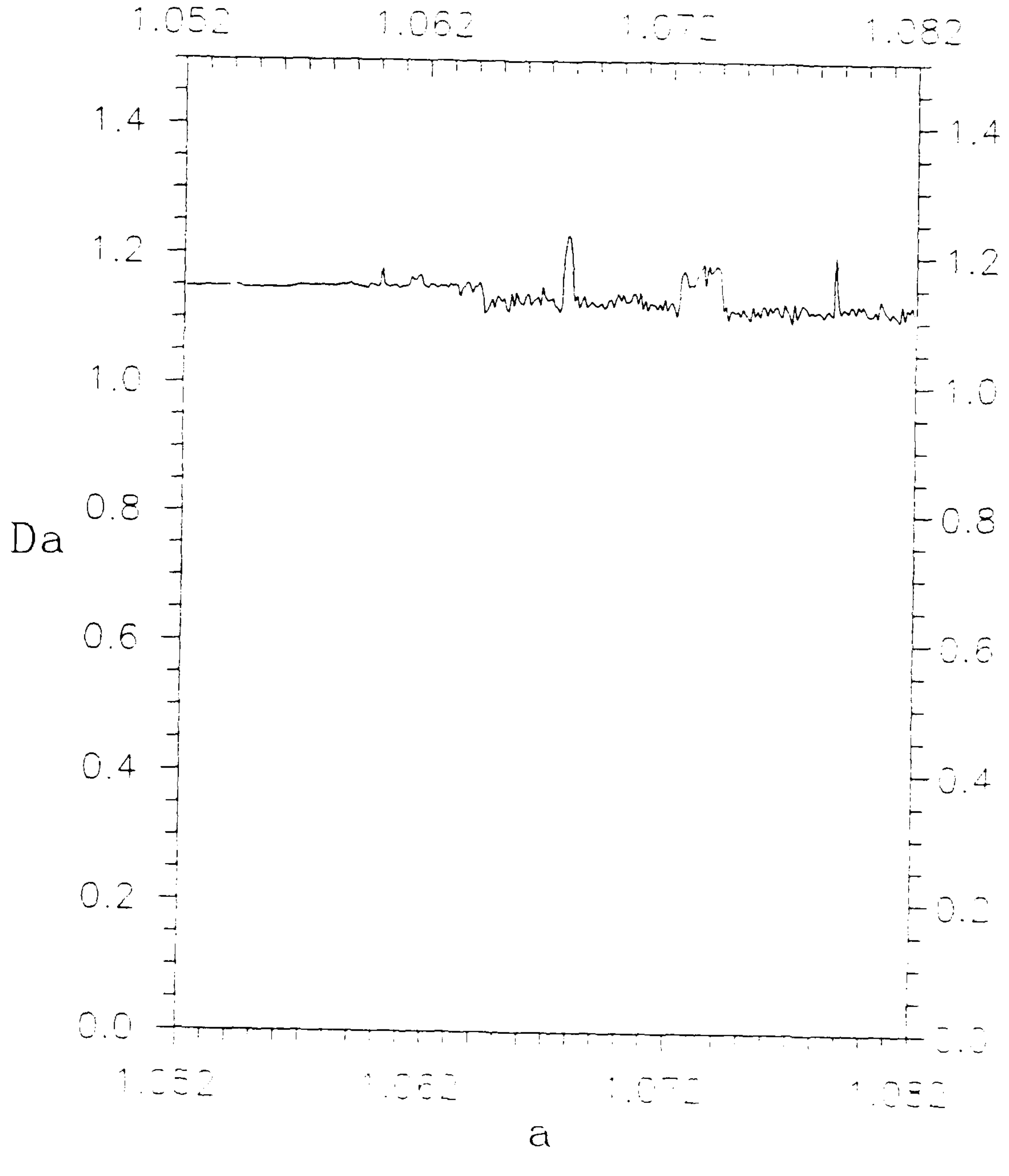

Figure 29. Fractal Dimension (D_A) Analysis of Henon Function, testing Δr with $b = 0.3$.

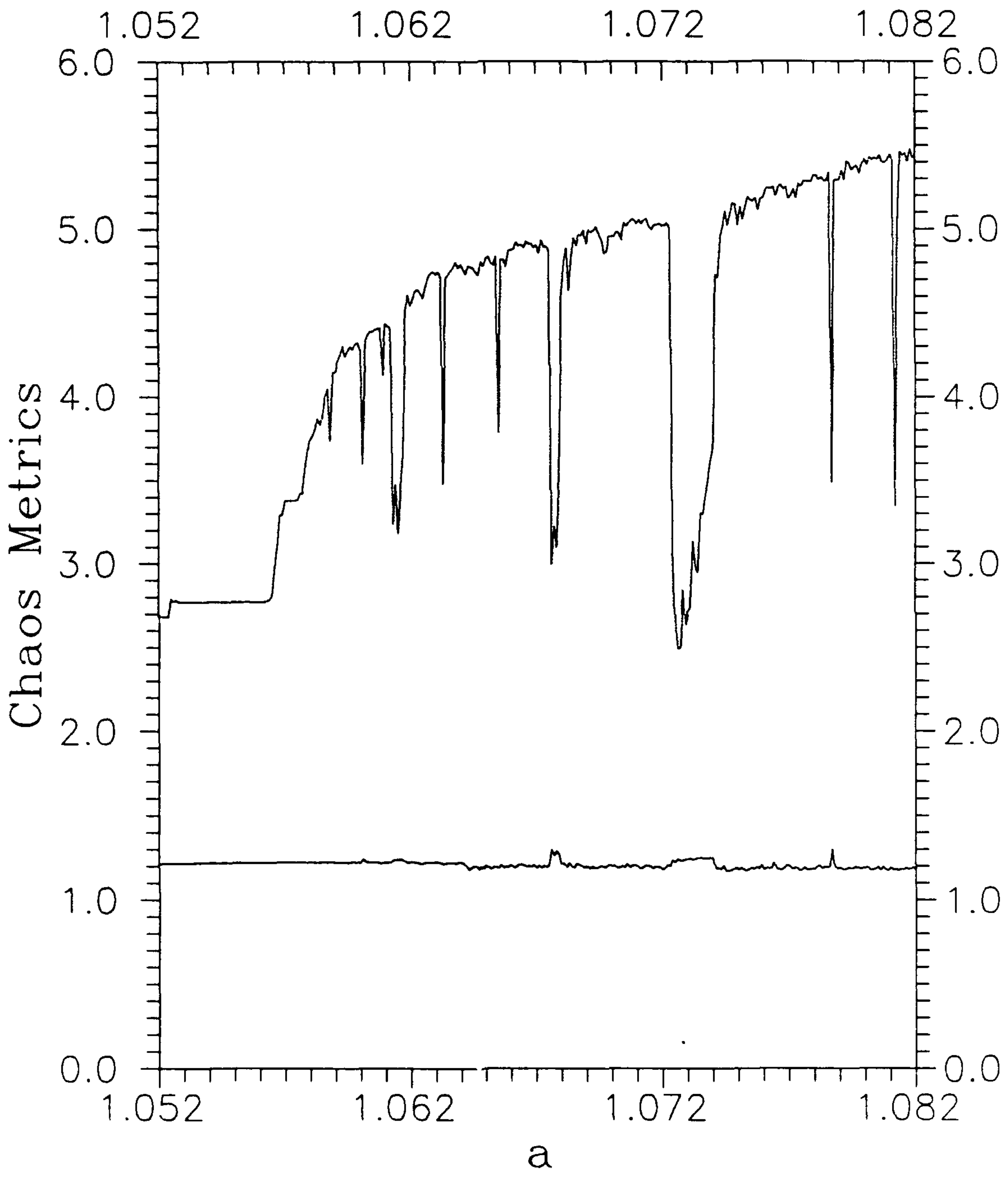

Figure 30. Fractal Dimension (D_A) Analysis of Henon Function, testing Δx, with b = 0.3.

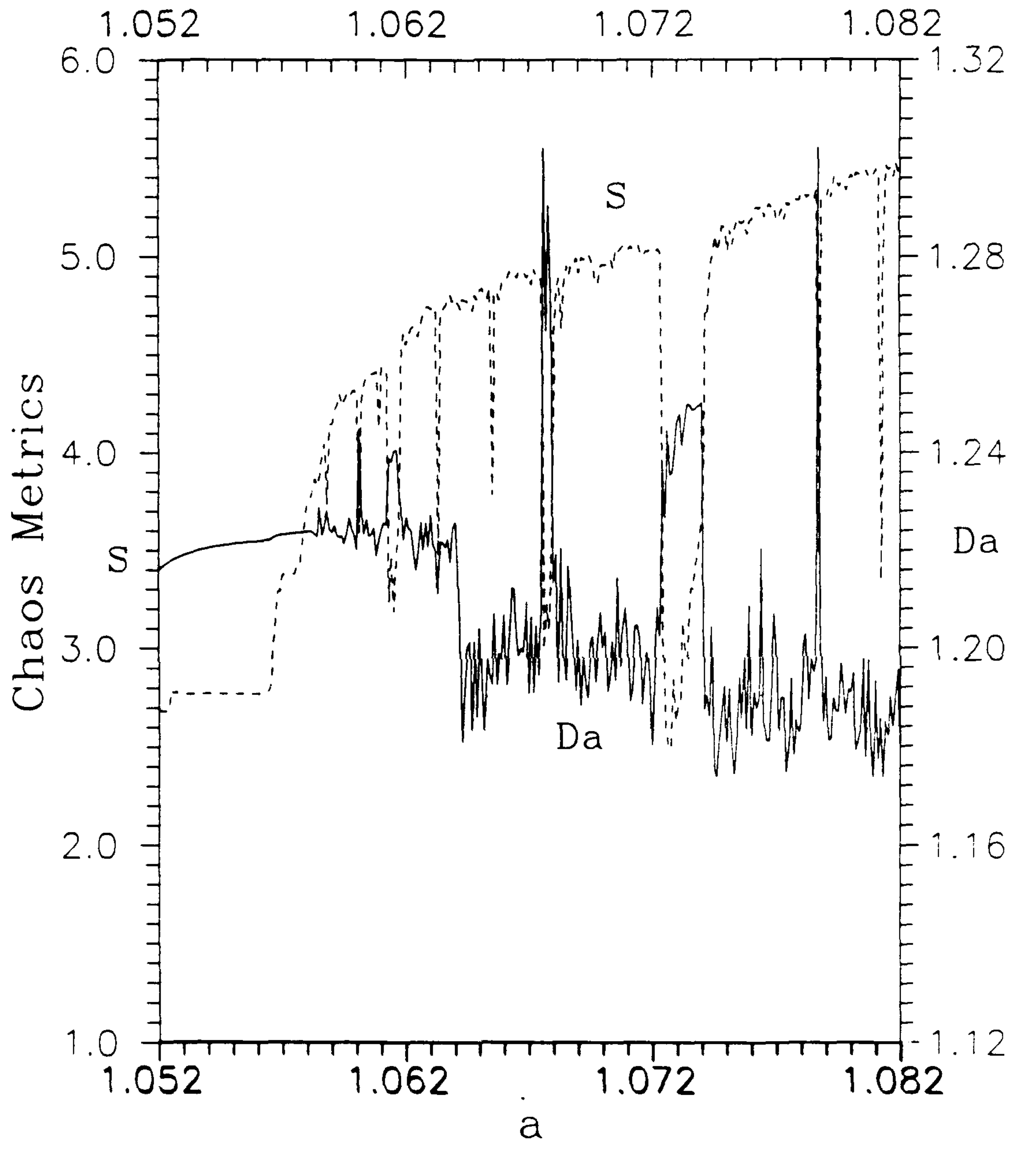

Figure 31. Fractal Dimension (D_A) and Entropy (S) Analysis of Henon Function, with b = 0.3.

behavior for high values of b, where both D_A and S show sudden increases in value.

4. Conclusions drawn from Logistics Difference and Henon equation analysis

The self-affine fractal dimension, D_A, is a good discriminator of low frequencies in data, e.g., periods 2, 4, and 8 of a given length data set. However, at high frequencies, period 16 and above in the initial bifurcation, it is difficult to use D_A to differentiate between periodicity and turbulence or chaos. The entropy, S, seems related to D_A, but S measures the evenness of particle position distribution, while D_A measures the changing apparent jaggedness of the function as the resolution varies. A sharp change in S always accompanies a sharp change in D_A, but not always vice versa. Therefore, D_A and S generally show the same trends within the chaos region, but not always. In such cases diffusion and dispersion rates are not equivalent.

The Lyapunov exponent provides perhaps the most definitive information: for a given value of λ, we know for certain whether the function is stable, periodic, or chaotic. Additionally, the Lyapunov exponent should be able to describe the degree of chaos: the greater the value of λ, the faster the orbital trajectory diverges, and therefore the more chaotic the function. Unfortunately, the Lyapunov exponent is not readily determined for multi-dimensional systems.

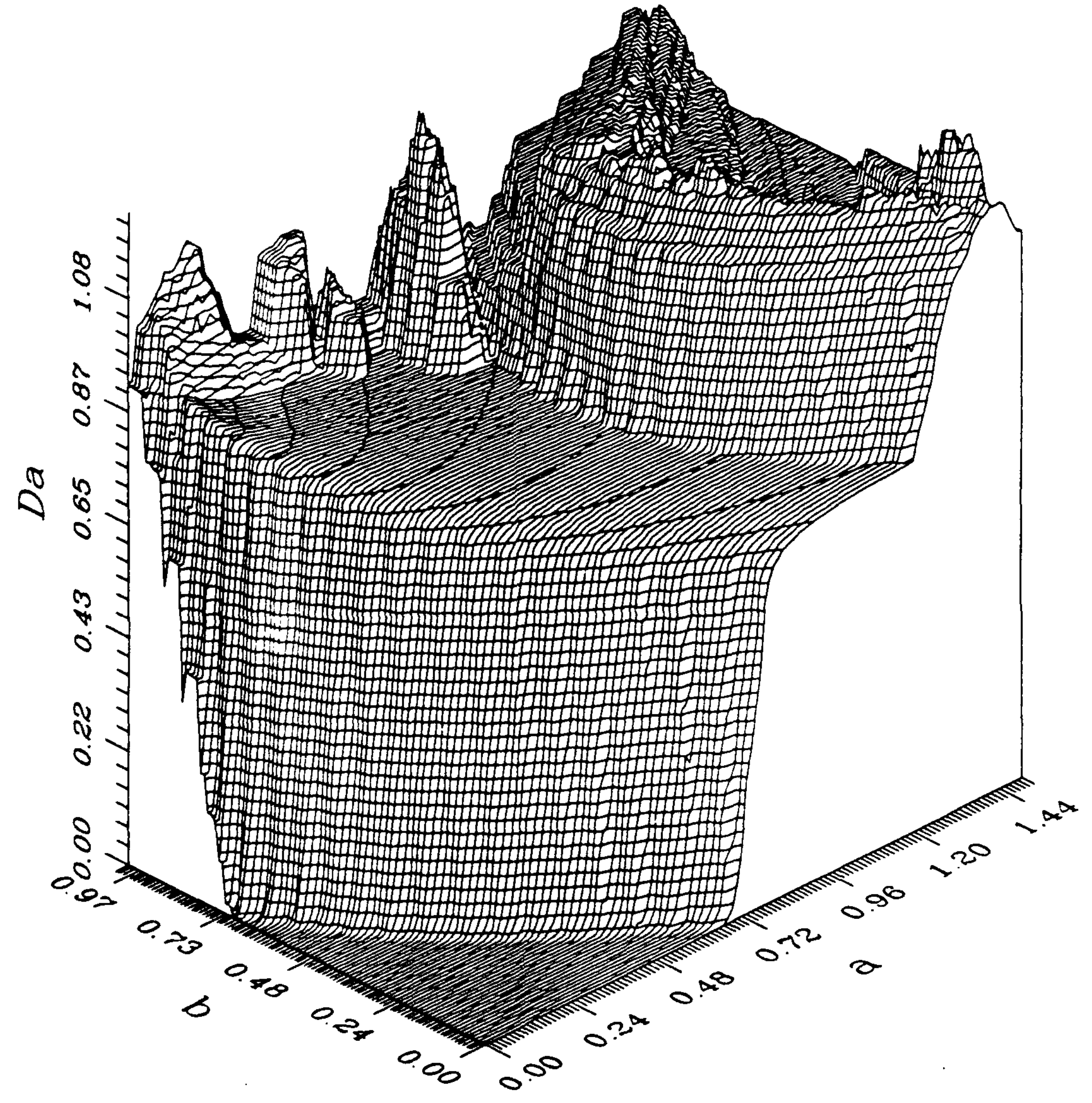

Figure 32. Fractal Dimension (D_A) Analysis of Henon Function, $x_{n+1} = 1-ax_n^2+y_n$, $y_{n+1} = bx_n$.

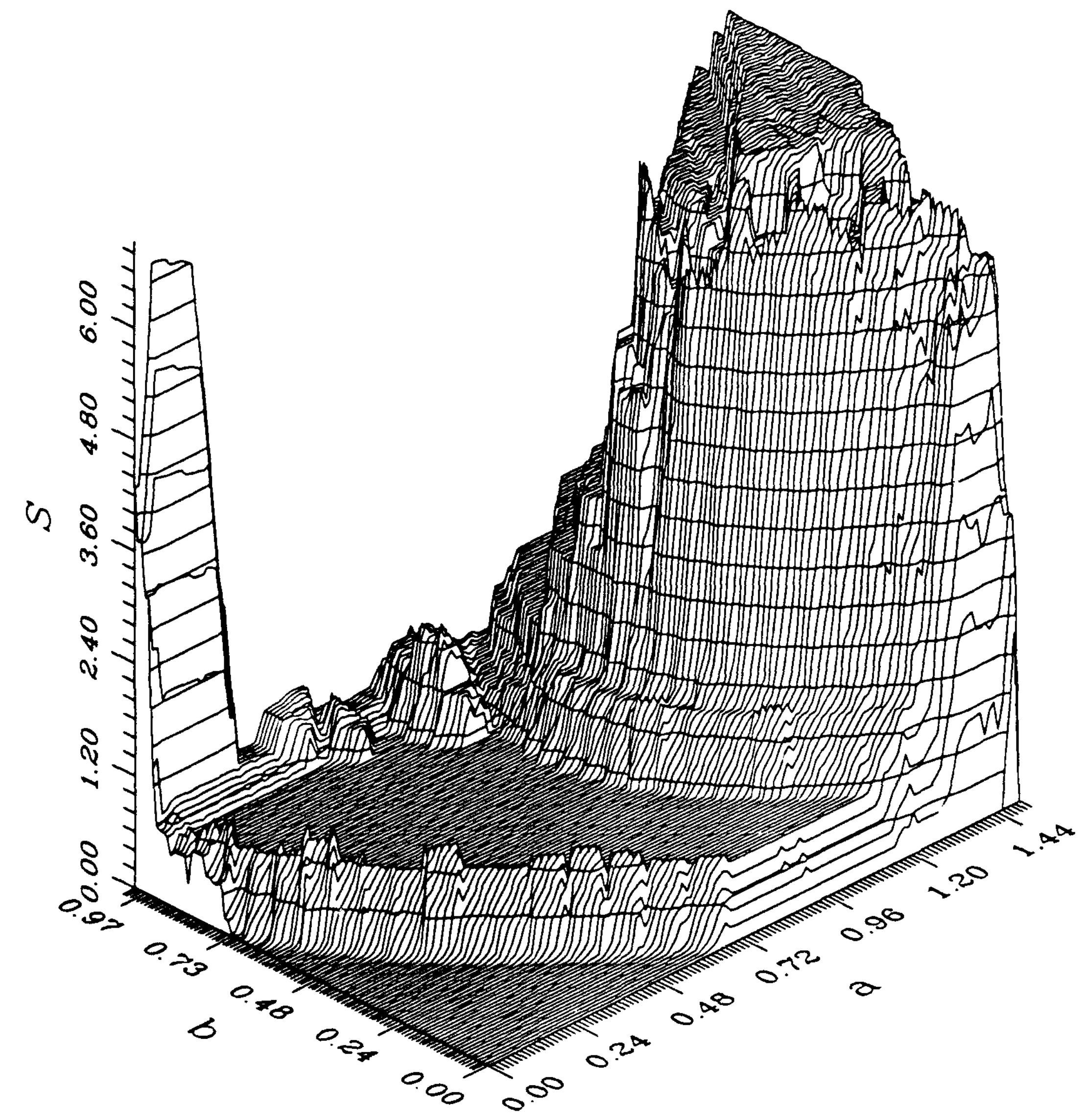

Figure 33. Entropy (S) Analysis of Henon Function, $x_{n+1} = 1-ax_n^2-y_n$, $y_{n+1} = bx_n$.

Analysis of these two simple systems, the logistics difference and Henon sets, shows that S, D_A, and λ display correlated but not identical behavior, since they measure rather different things. These insights can now be applied in studying atmospheric Lagrangian particle models.

C. MCNIDER PARTICLE DISPERSION MODEL ANALYSIS

1. Methodology

Four aspects of the McNider Particle Dispersion Model were studied: the divergences of vertical position, total position, velocity, and phase velocity. The position was measured in radial distance from the particle origin, which was arbitrarily set at x,y = 0, z = 100 meters. To simulate the real atmosphere, the boundary layer inversion height was varied linearly from 2,000 to 200 meters as 1/L, the inverse Monin-Obukhov length, was varied from -0.2 to 0.1. Mean velocities were set to zero and total particle velocity reflection was assumed at the ground surface and inversion.

Hints as to model behavior are again provided by the bifurcation maps, which now depict the expansion and distribution of particle range with decreasing stability, as measured by 1/L. Model performance was checked against standard geophysical measures such as the Brunt Vaisala Frequency (BVF), turbulent kinetic energy (tke), and vertical velocity variance (σ_w^2), as well as against the two readily calculated chaos measures, entropy, S, and the self-affine fractal dimension, D_A, for the range of 1/L values seen in the atmosphere. Using 100 partitions, the computed maximum possible entropy for all plots in the analysis was

$$S_{\max} = 4.6 \quad . \qquad \text{(III-5)}$$

For this study 1/L was set at a particular value, a single particle was released and followed for 3,600 time increments of 1/6 seconds each (600 seconds total), then 1/L was incremented and the analysis repeated, until the entire range was spanned using 1/L increments of 0.01.

2. McNider No-Skew Routine

For real unstable atmospheres, the vertical velocity distribution is negatively skewed [Figure 10]. Updrafts have higher velocities and thus occupy less volume than downdrafts. However, for the initial analysis of the McNider model, the vertical velocity turbulence distribution was not skewed. This was done to check the model without the ad hoc method McNider developed to introduce skewness, and which as outlined in Pielke (1984, pp. 178-179) appears to be incorrect.

An r versus 1/L plot of the particle position shows that on the negative (unstable) side, D_A and S follow roughly the same trend up to 1/L = 0 [Figure 34]. From the analysis of D_A in the Henon system, this suggests that the ratio of large to small-scale vertical distances between points varies little in the unstable regime without overlaying skewness. However, D_A again jumps suddenly when 1/L exceeds zero. Since vertical fluctuations tend to decay in general with increasing stability, one expects that S would decay and D_A would tend upward steadily from left to right for real turbulence in the atmosphere. Again, the discontinuities near $1/L = 0$ are probably due to discontinuities across the neutral transition in the McNider algorithms. The small scale fluctuations in S with 1/L are probably due to the random nature of the particle paths.

On the positive (stable) side of 1/L, both D_A and S show sudden dramatic step increases, followed by D_A tending upward while S tends downward. Again, this highlights the fact that S and D_A are not measuring the same thing. The sudden jumps in S in the McNider model seem antithetical to the fact that negative buoyancy tends to suppress vertical motion and render it oscillatory in real fluids. For growing positive values of 1/L, the decrease in S shows that the distribution of points within the domain becomes less uniform, while the rise in D_A suggests that the distance between successive points decreases more slowly on the small scale than for the larger, longer time scale fluctuations. This is qualitatively consistent with measurements under increasingly stable conditions in real fluids (Stull, 1988).

The bifurcation diagram [Figure 35] also shows that for $1/L < 0$ the total range of positions steadily decreases with increasing stability, while for $1/L > 0$ it hardly varies.

This is because the total position change includes both vertical **and** horizontal fluctuations, and horizontal fluctuations are much less dependent on stability.

These plots reveal other weaknesses in the McNider model. Unrealistically sharp fluctuations seem to occur near the neutral stability transition ($1/L = 0$), probably due to the dichotomous nature of the McNider algorithms, one set for stable conditions, another for unstable. From equation II-18, it is also clear for isotropic turbulence that $\sigma_w^2 = 2/3$ tke. Stability suppresses vertical turbulence, while instability enhances it. So for $1/L < 0$, σ_w^2 should exceed 2/3 tke. Figure 34 shows increasingly qualitatively incorrect ratios with increasing instability. σ_w^2 also drops to near zero at neutral stability ($1/L \cong 0$) due to very low mean windshear values in the mesoscale flow simulation. This may not be as significant a problem in the prognostic windflow models for which the McNider model was originally designed as it is in the similarity-based flow model simulator used in this study. However, an artificial "dip" would probably still appear.

The bifurcation map of the total 3-D position seems to show a fairly constant range for $1/L > 0$, a jump at zero, and a varying range for $1/L < 0$ [Figure 35]. The plot density is too heavy to discern actual distribution patterns for given values of $1/L$, although the computer screen displays a Moire-like pattern. What can be extracted from this plot is the

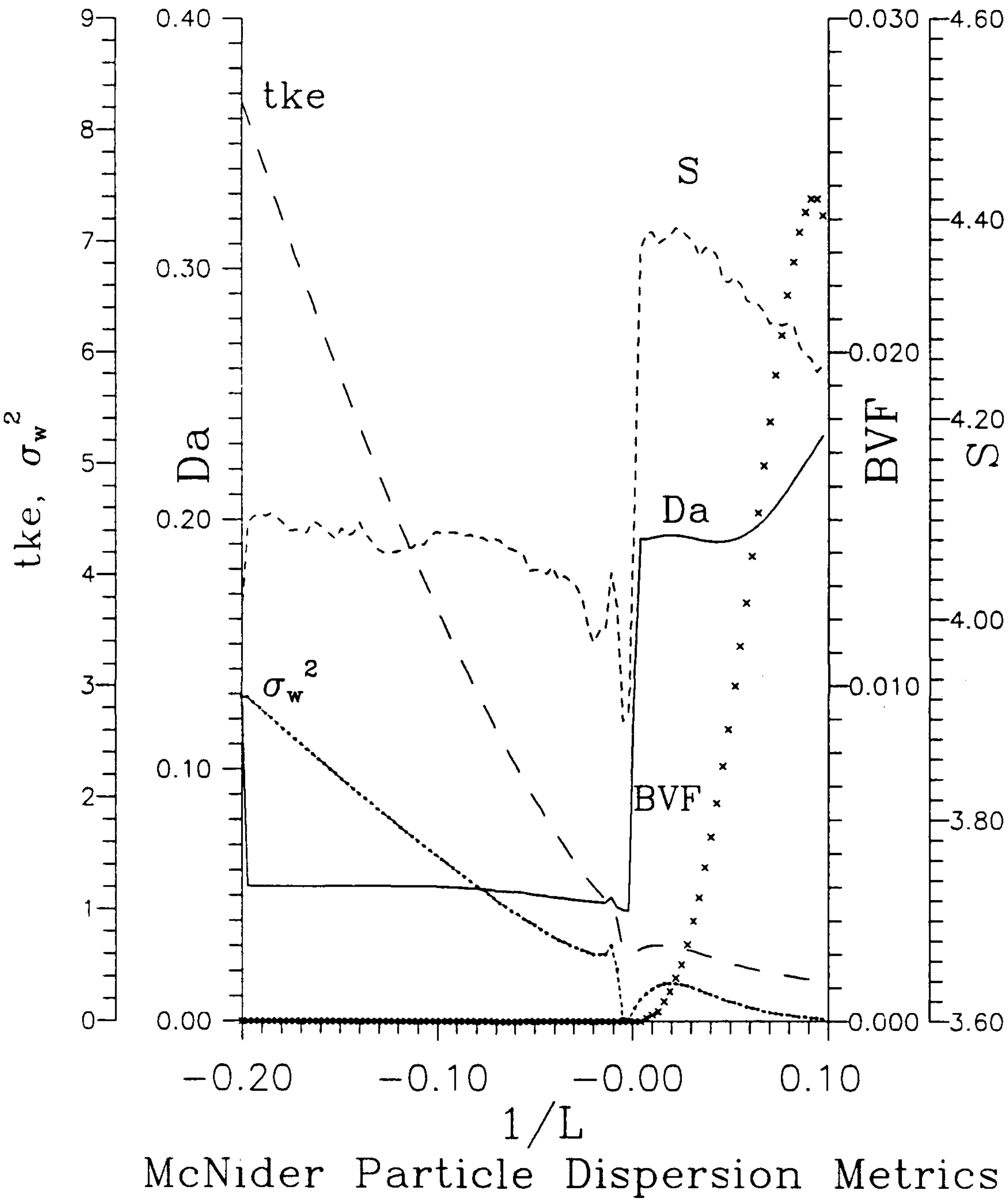

Figure 34. McNider Particle Dispersion Model. Test of total 3-D distance, Δr, with no skew.

general density of points and the upper and lower boundaries (in magnitude) of position.

In contrast to Figure 34, a plot of only the vertical position change (Δz) reveals that the pattern for D_A is quite similar to that for S [Figure 36]. The bifurcation map shows an opposite jump in the vertical position as 1/L crosses the zero-point [Figure 37].

The velocity distribution for various values of 1/L seems fairly constant, so that the S curve has only a slight upward trend [Figure 38], except at values near $1/L = 0$. However, D_A tends downward strongly from left to right, jumps at zero, then again tends downward at around $1/L = 0.7$, where it begins increasing in value. This upward trend in D_A is probably caused by increasing negative buoyancy. The bifurcation map [Figure 39] shows that the velocity range tends generally downward, with a jump near $1/L = 0$, and a change in trend at $1/L = 0.7$.

The phase velocity plots are almost identical to the total position plots. This is because in calculation phase velocity the velocities are small in magnitude compared to the position values.

3. McNider Model, Skew On

A glance at Figure 42 shows a big problem with the vertical velocity skewness algorithm. The tke and σ_w^2 values are much too large. An expected maximum σ_w^2 will be ~ $10 m^2/s^2$,

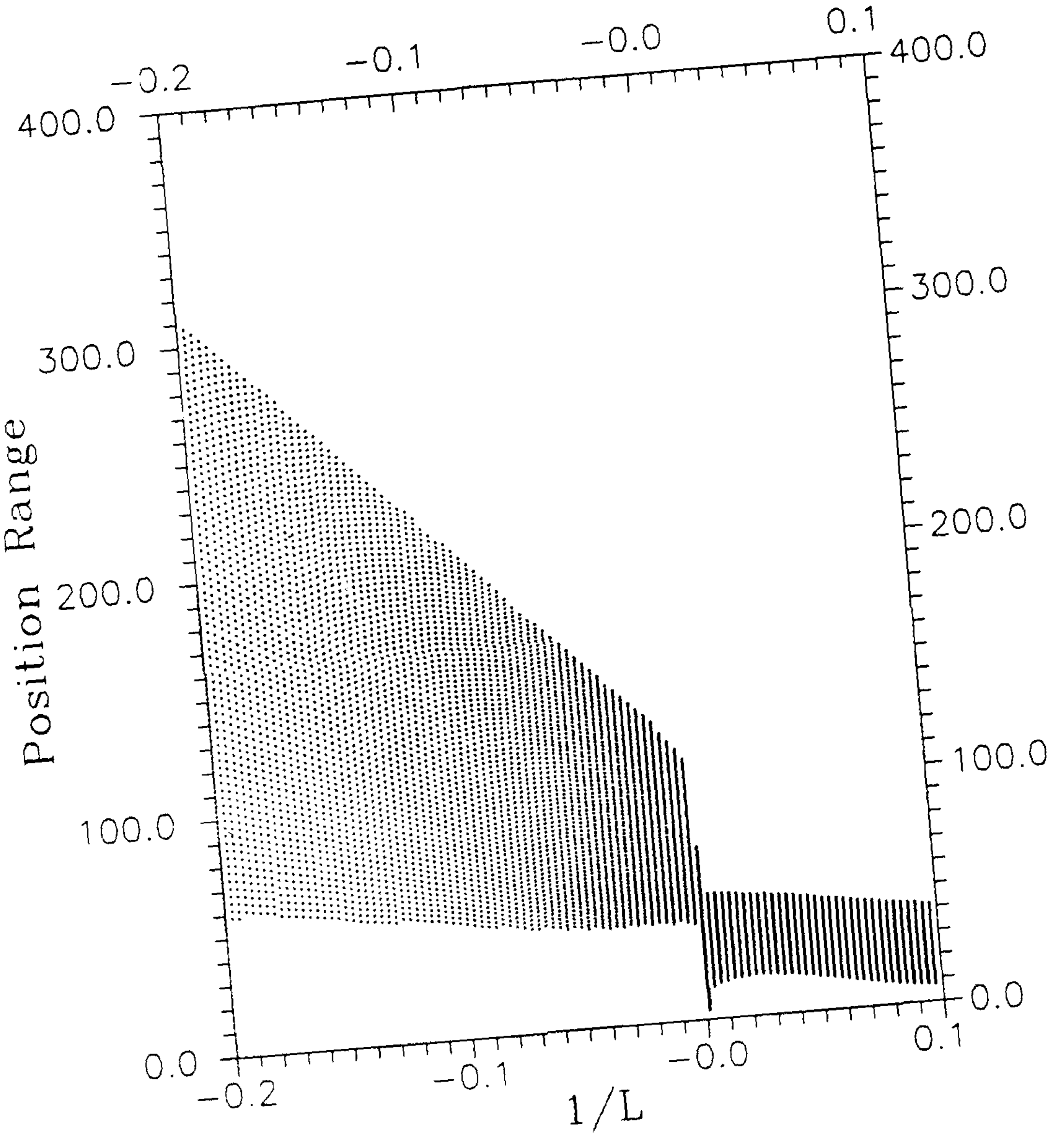

Figure 35. McNider Particle Dispersion Model Bifurcation Map. Test of total 3-D distance, Δr, with no skew.

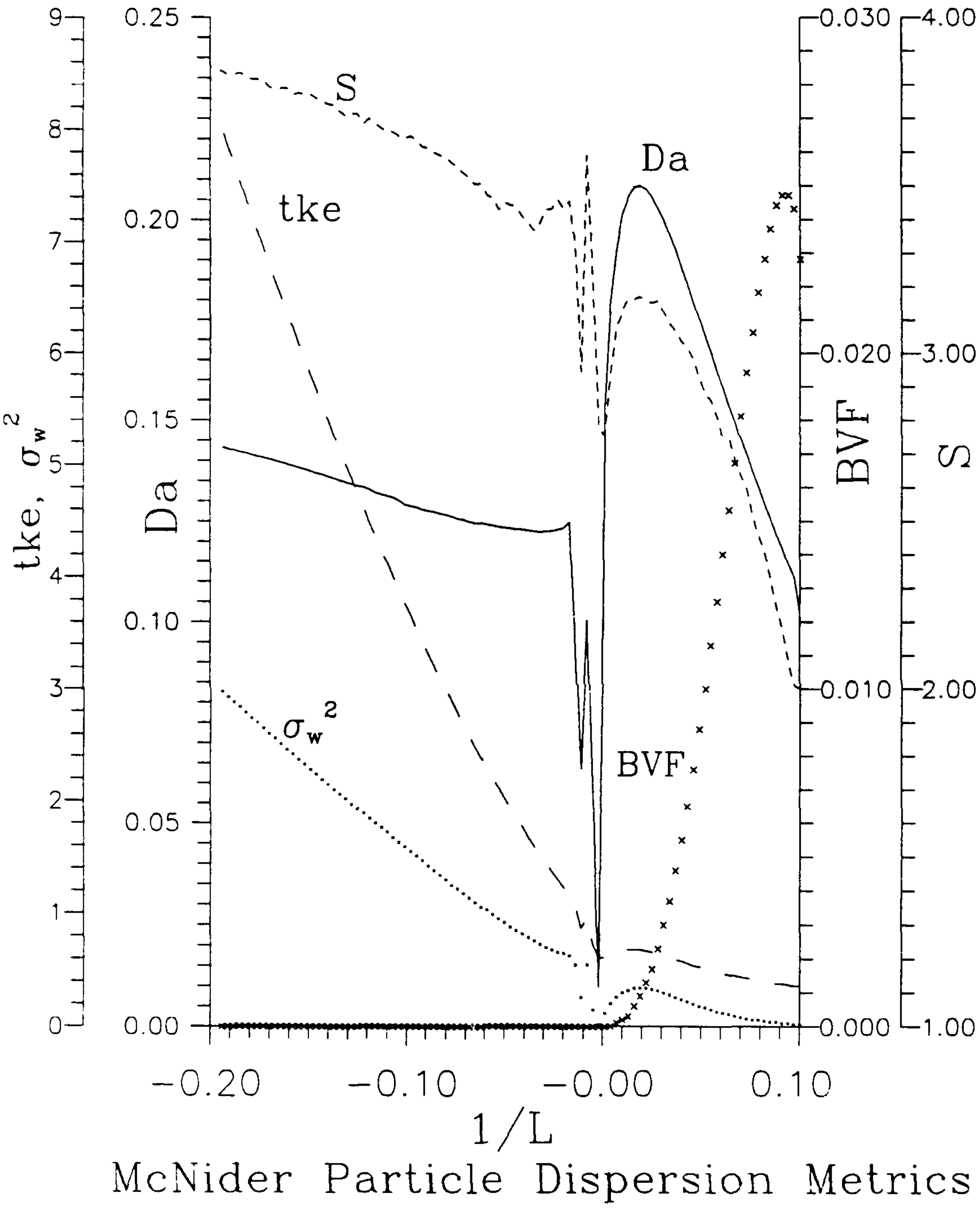

Figure 36. McNider Particle Dispersion Model. Test of vertical position, Δz, with no skew.

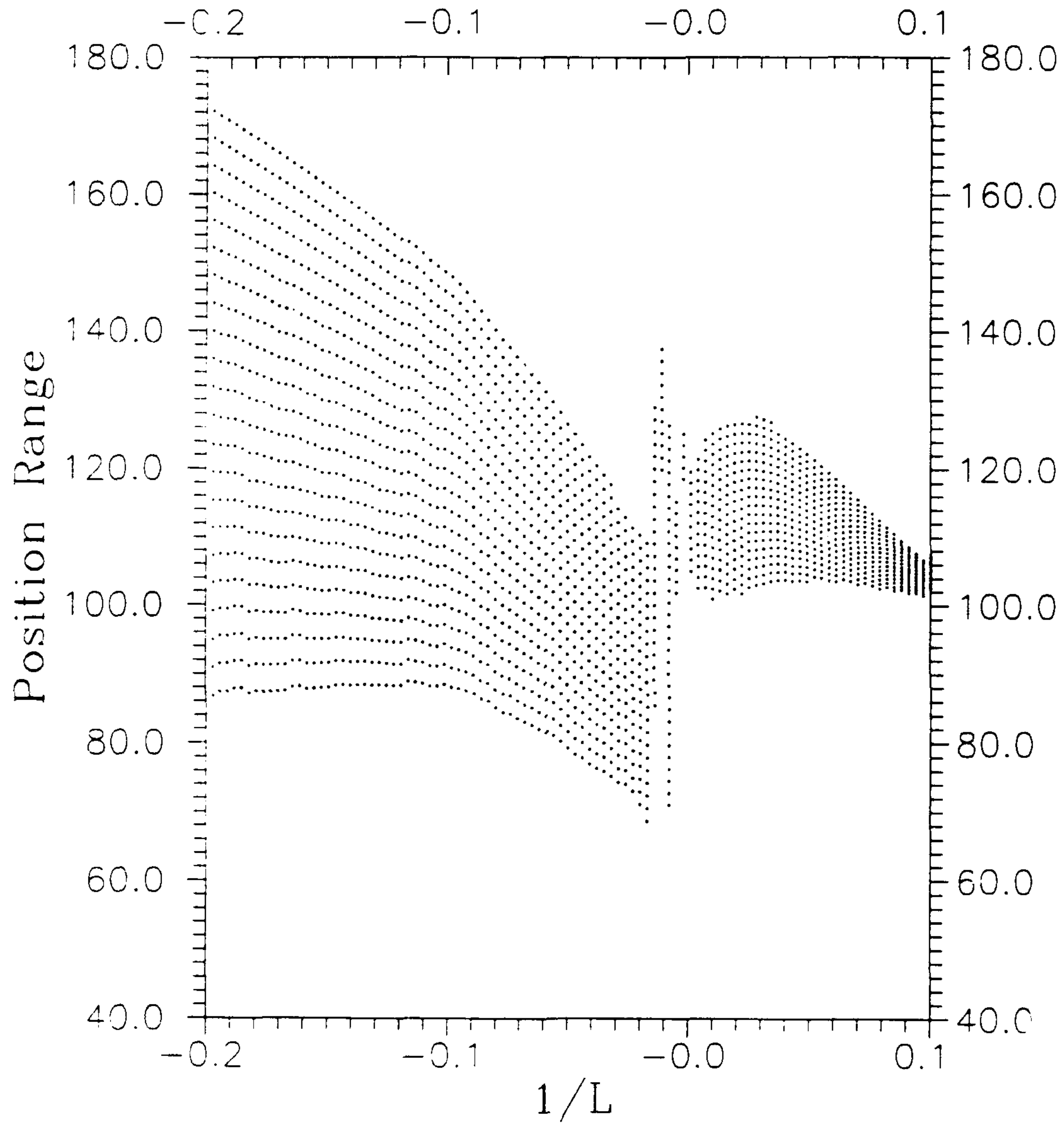

Figure 37. McNider Particle Dispersion Model Bifurcation Map, Test of vertical distance, Δz, with no skew.

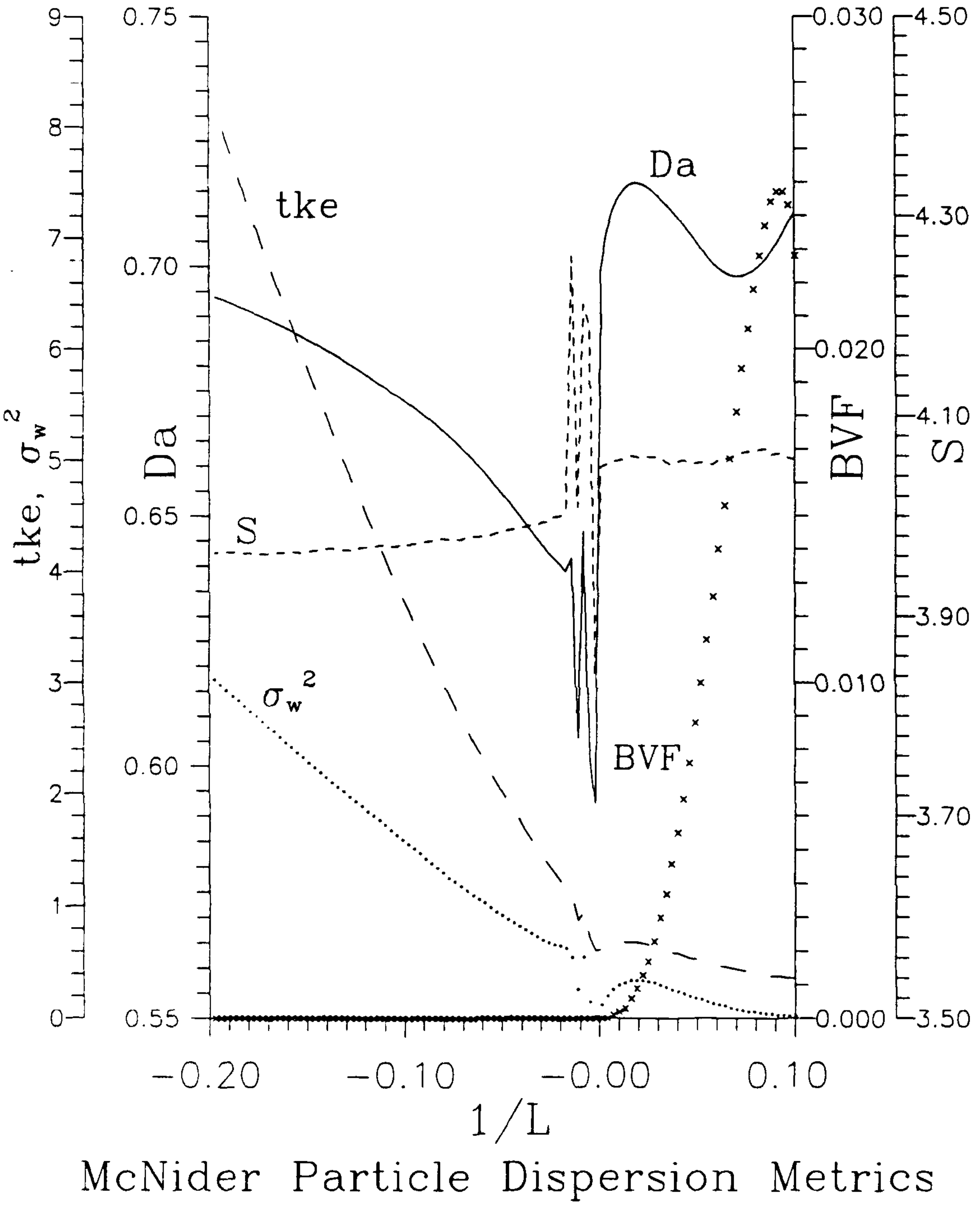

Figure 38. McNider Particle Dispersion Model, Test of total velocity, no skew.

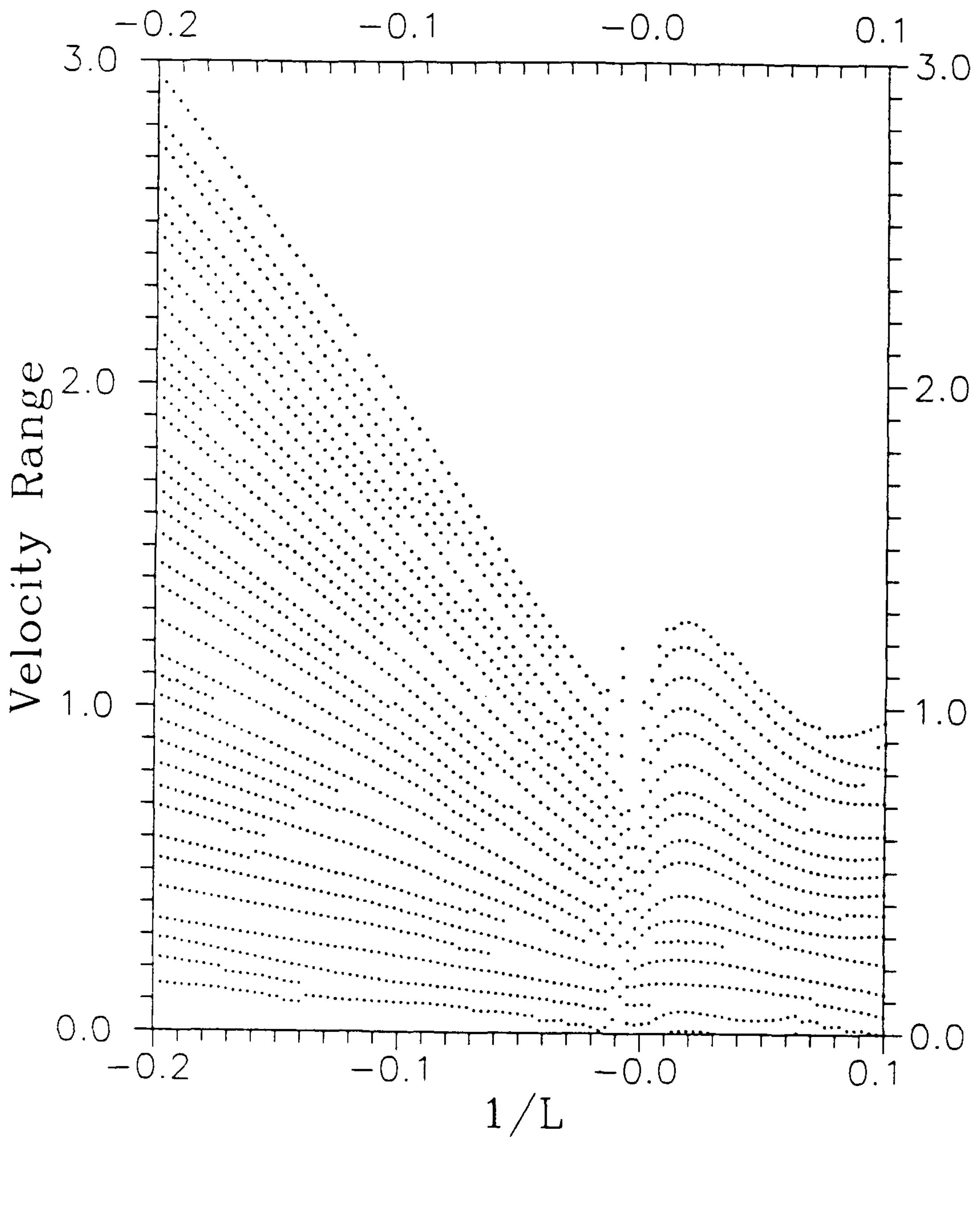

Figure 39. McNider Particle Dispersion Model Bifurcation Map. Test of velocity, with no skew.

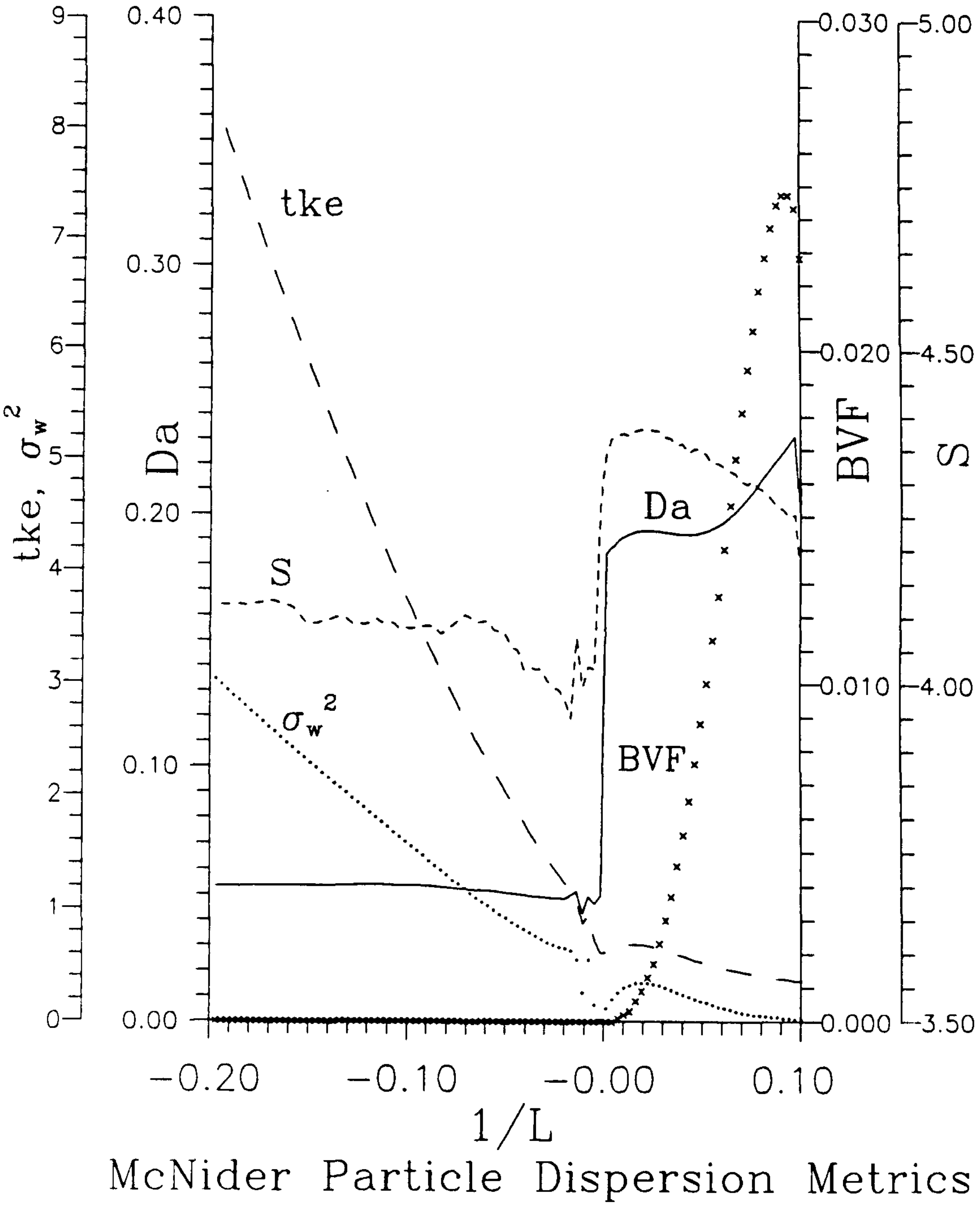

Figure 40. McNider Particle Dispersion Model. Test of Phase Velocity, with no skew.

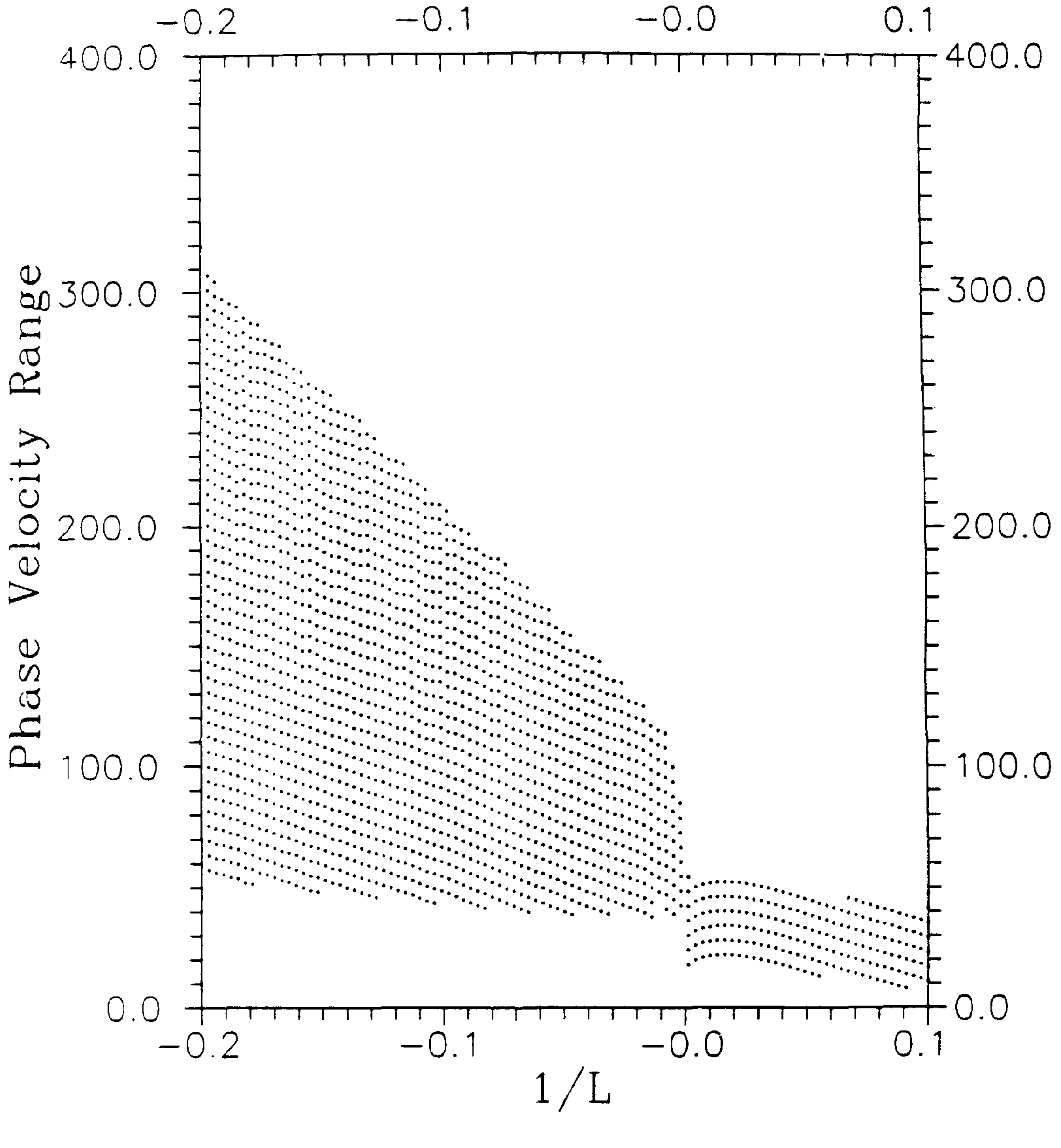

Figure 41. McNider Particle Dispersion Model Bifurcation Map. Test of phase velocity, with no skew.

while the plot shows ~ 100 m^2/s^2. It seems that the algorithm is incorrect for very unstable values of 1/L. However, the stable side (positive values of 1/L) shows well behaved, reasonable values for tke and σ_w^2.

Examining only the stable side (positive 1/L) of Figures 42-49, D_A and S are surprisingly flat for velocity, but not for total 3-D position, vertical position change, or phase velocity. Also, the bifurcation plot of velocity range becomes quite small for positive 1/L (Figure 47). Oddly enough, the small values begin not at $1/L > 0$, but $1/L > 0.02$. This corresponds to a sharp drop in entropy and fractal dimension near this value of 1/L, suggesting that the skewness algorithm is responsible. However, re-examining Figures 36 and 38 with "no-skew", entropy and D_A drop sharply at $1/L = 0.02$ as well as at $1/L = 0$, suggesting that something other than the skewness algorithm is responsible, perhaps the inherent dichotomy in equations II-44 or II-37a.

Again, this sharp transition near neutral values of 1/L does not seem to accurately reflect nature, but rather appears to be a discontinuity in the algorithm solutions.

Figure 47 also shows that the velocity range is much too high for the unstable case; an expected velocity value is on the order of 10 m/s or less. Oddly, there are several windows where the velocity range drops to low values, and are roughly what is expected, i.e., for $1/L \approx -0.475$. However, at

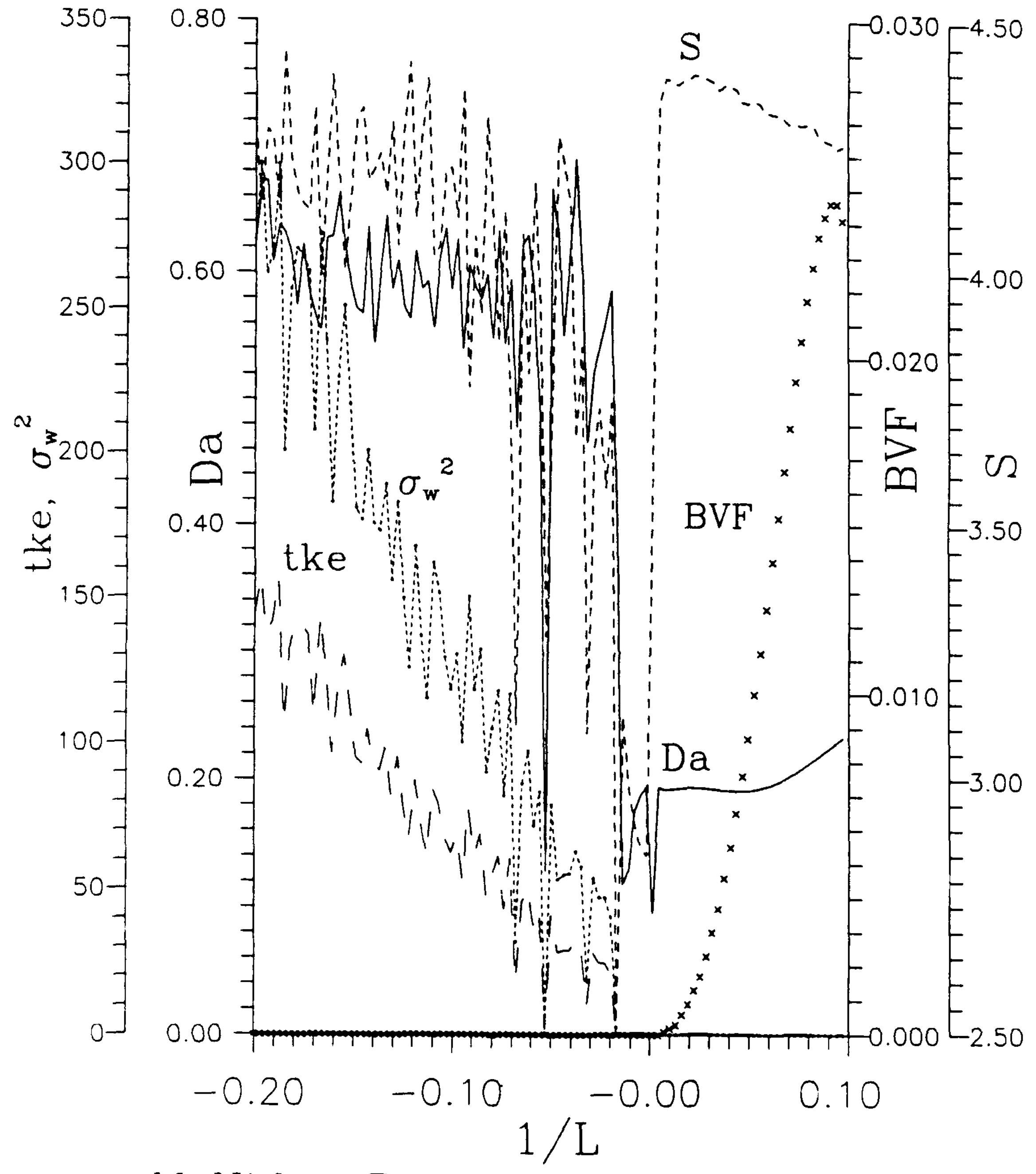

Figure 42. McNider Particle Dispersion Model. Test of total position, Δr, with skewness.

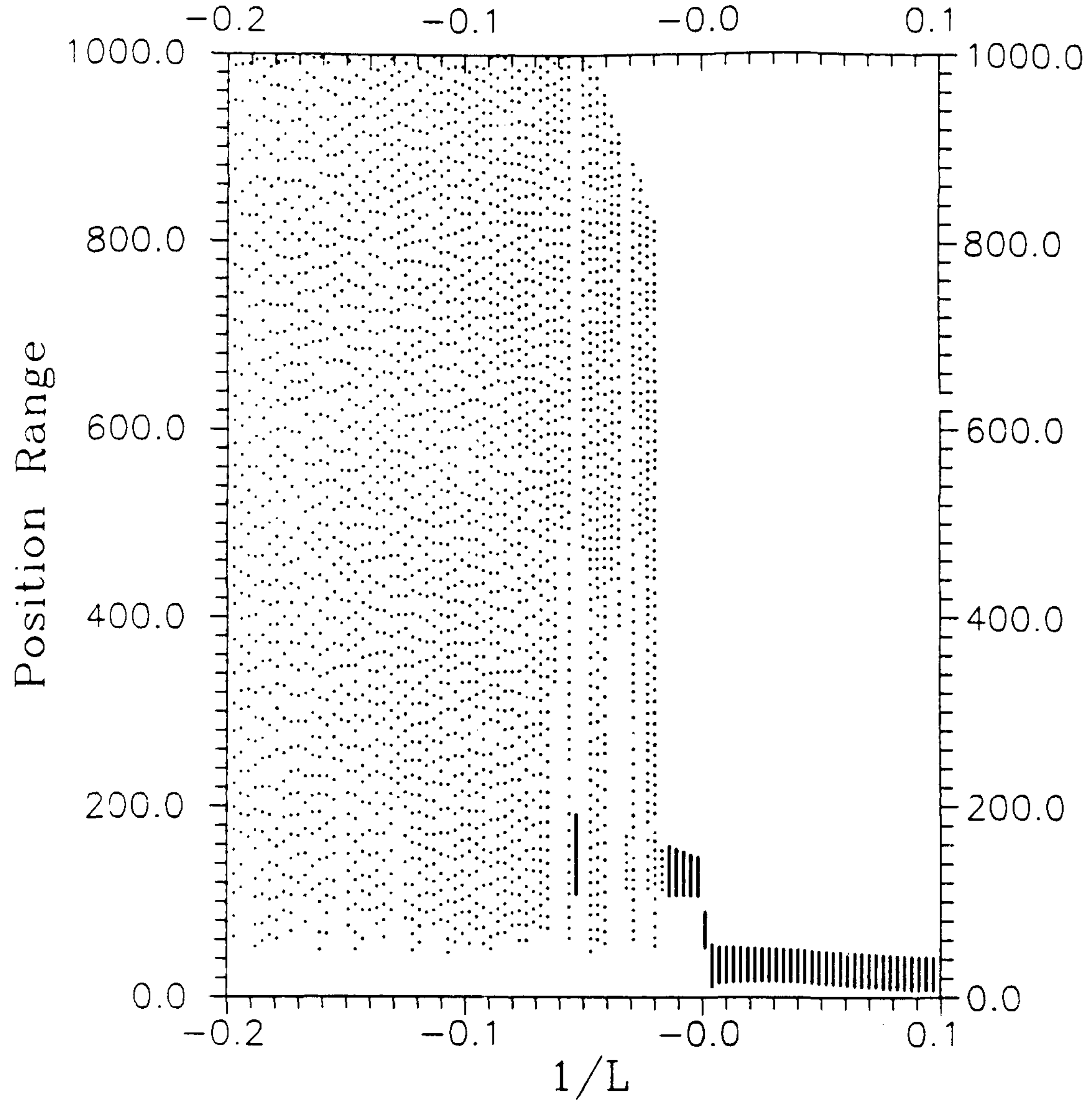

Figure 43. McNider Particle Dispersion Model Bifurcation Map. Test of tctal position, Δr, with skewness.

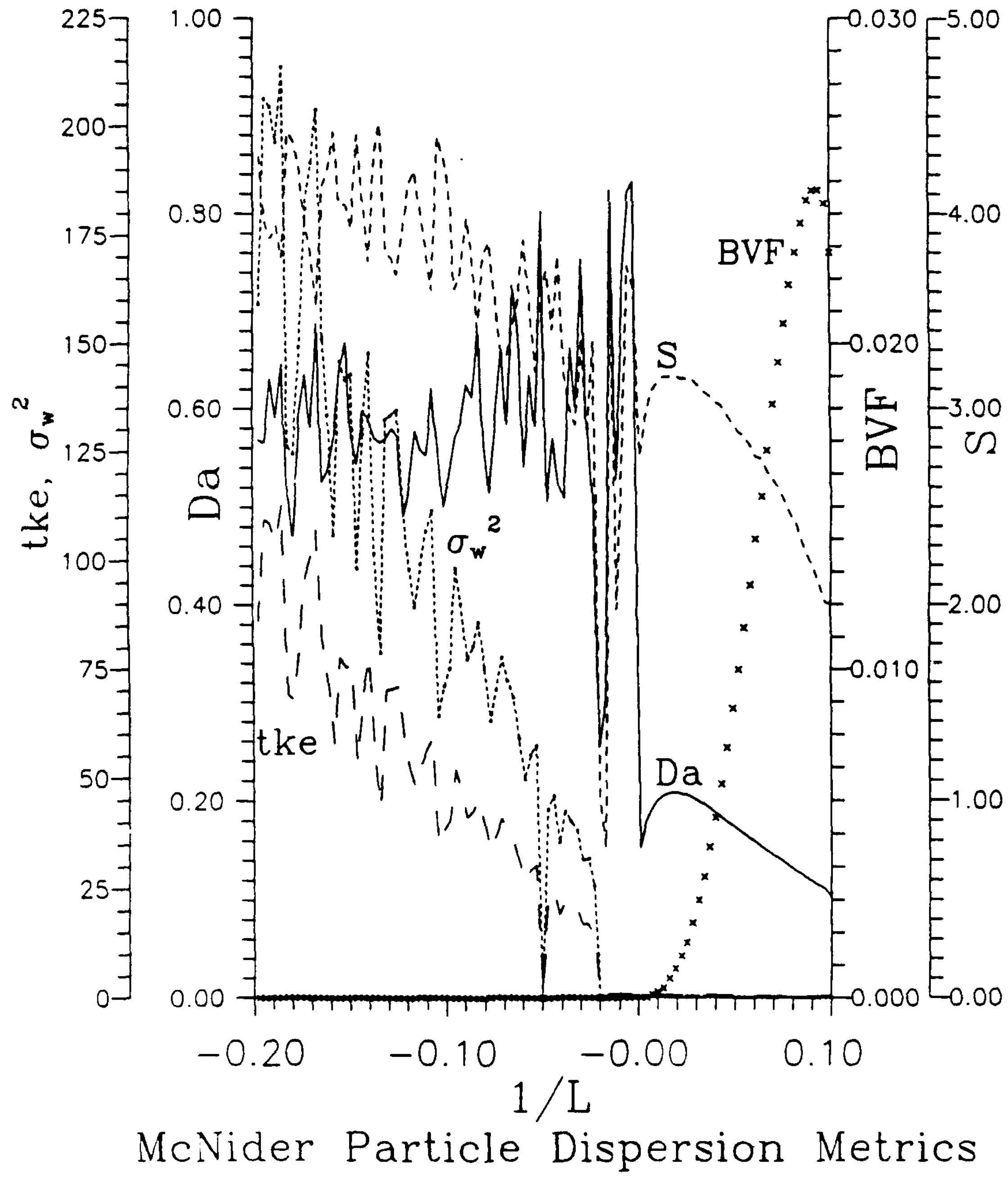

Figure 44. McNider Particle Dispersion Model. Test of vertical position, Δz, with skewness.

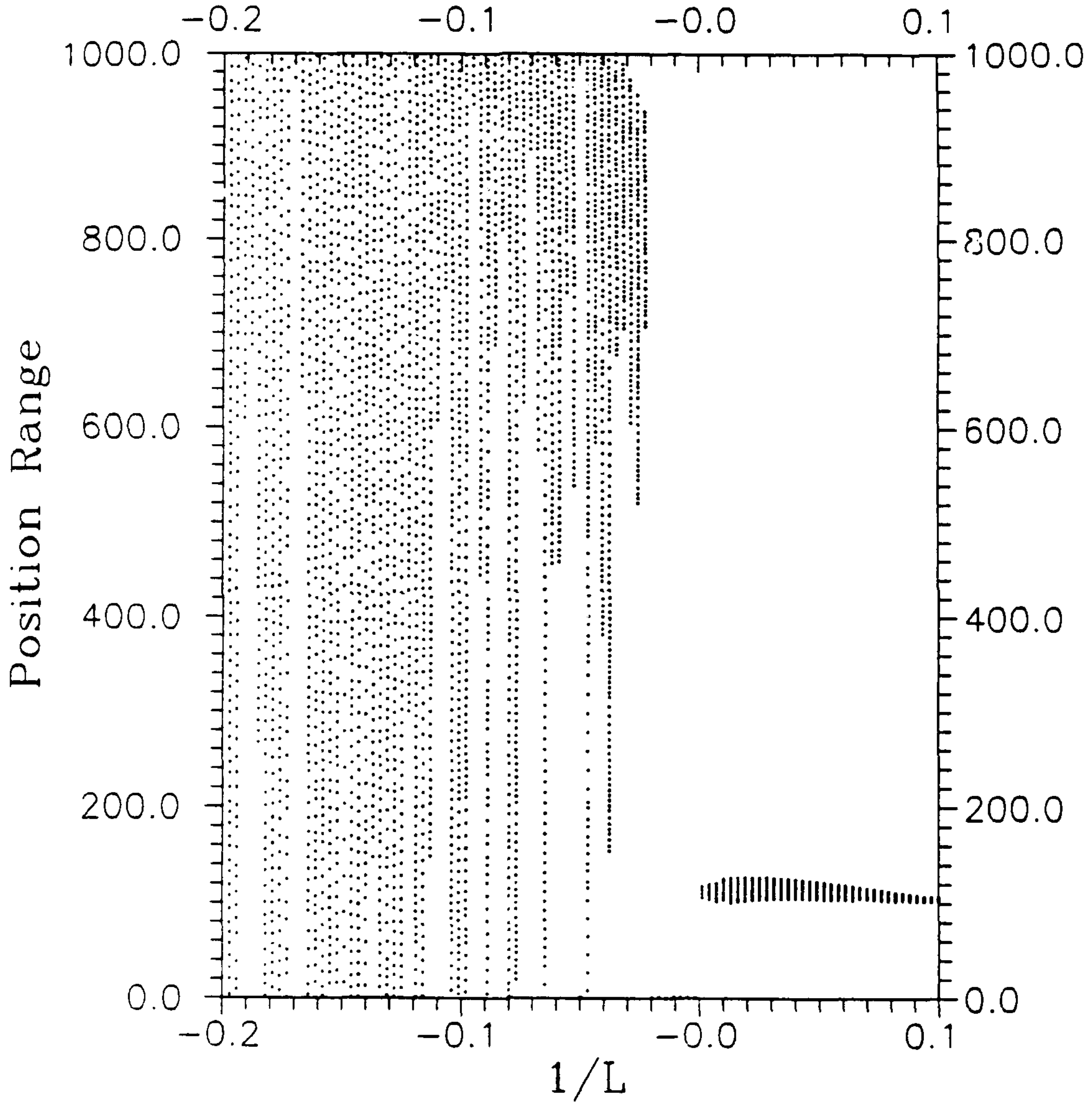

Figure 45. McNider Particle Dispersion Model Bifurcation Map. Test of vertical position, Δz, with skewness.

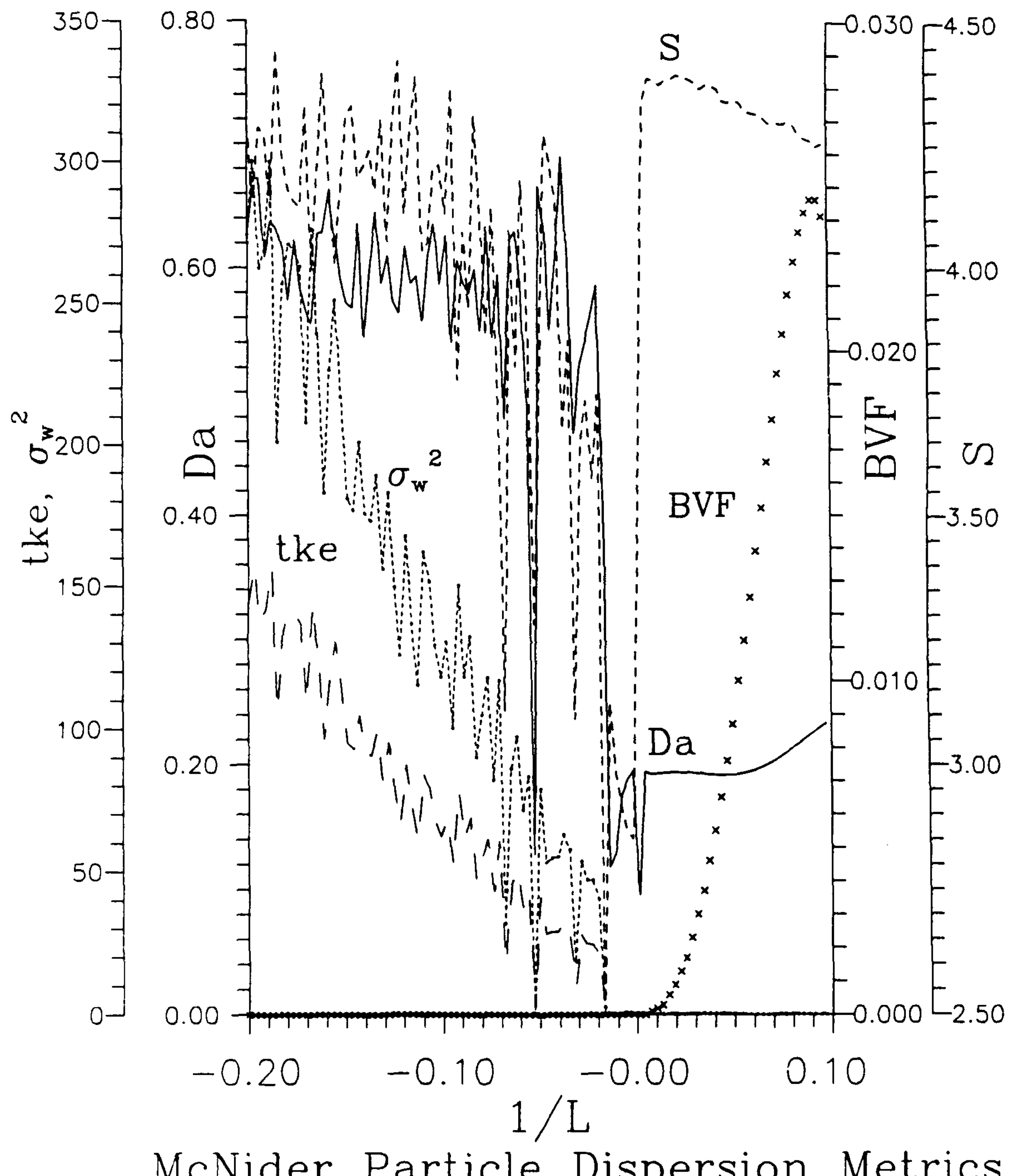

Figure 46. McNider Particle Dispersion Model. Test of total velocity, with skewness.

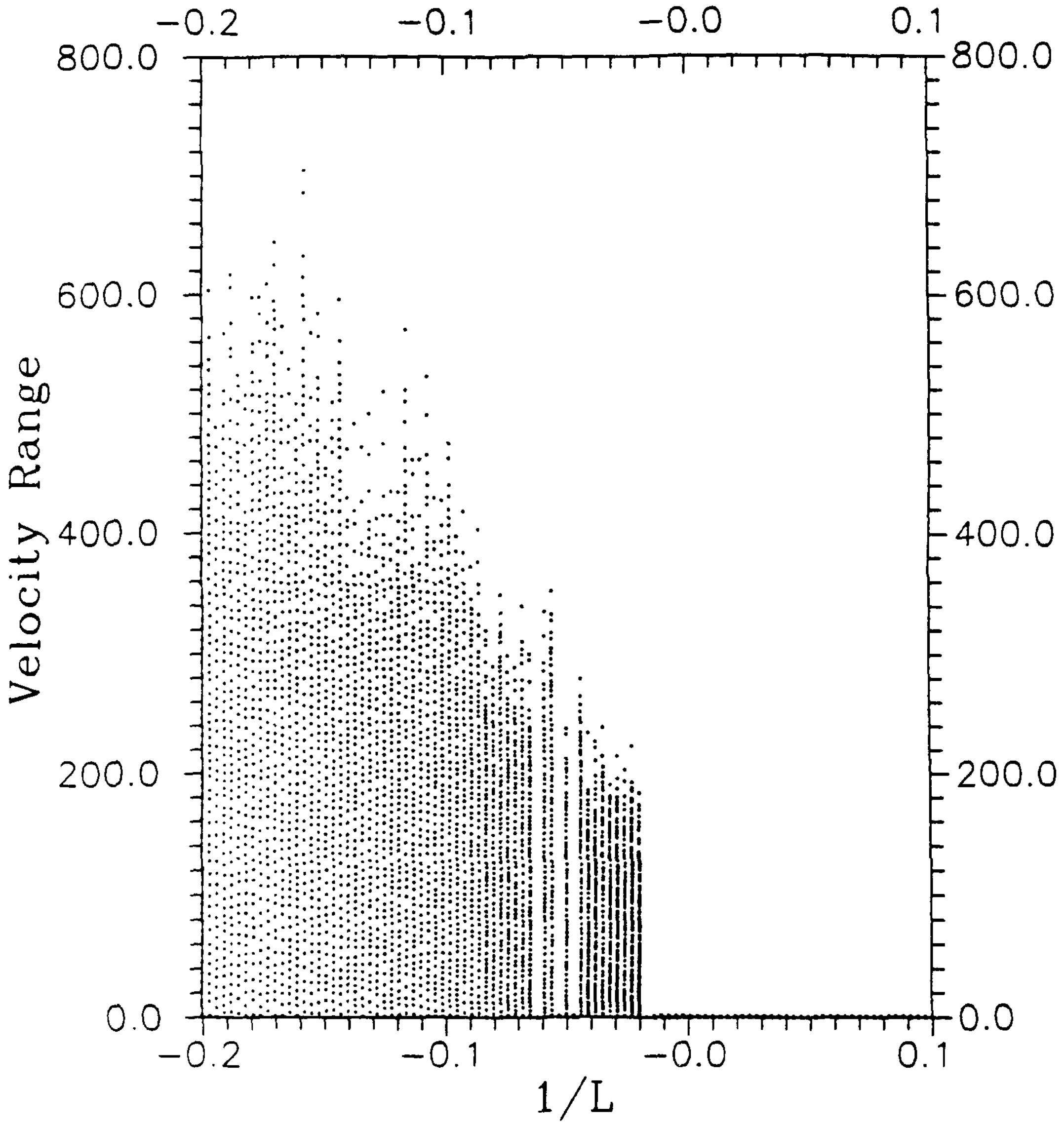

Figure 47. McNider Particle Dispersion Model Bifurcation Map. Test of total velocity, with skewness.

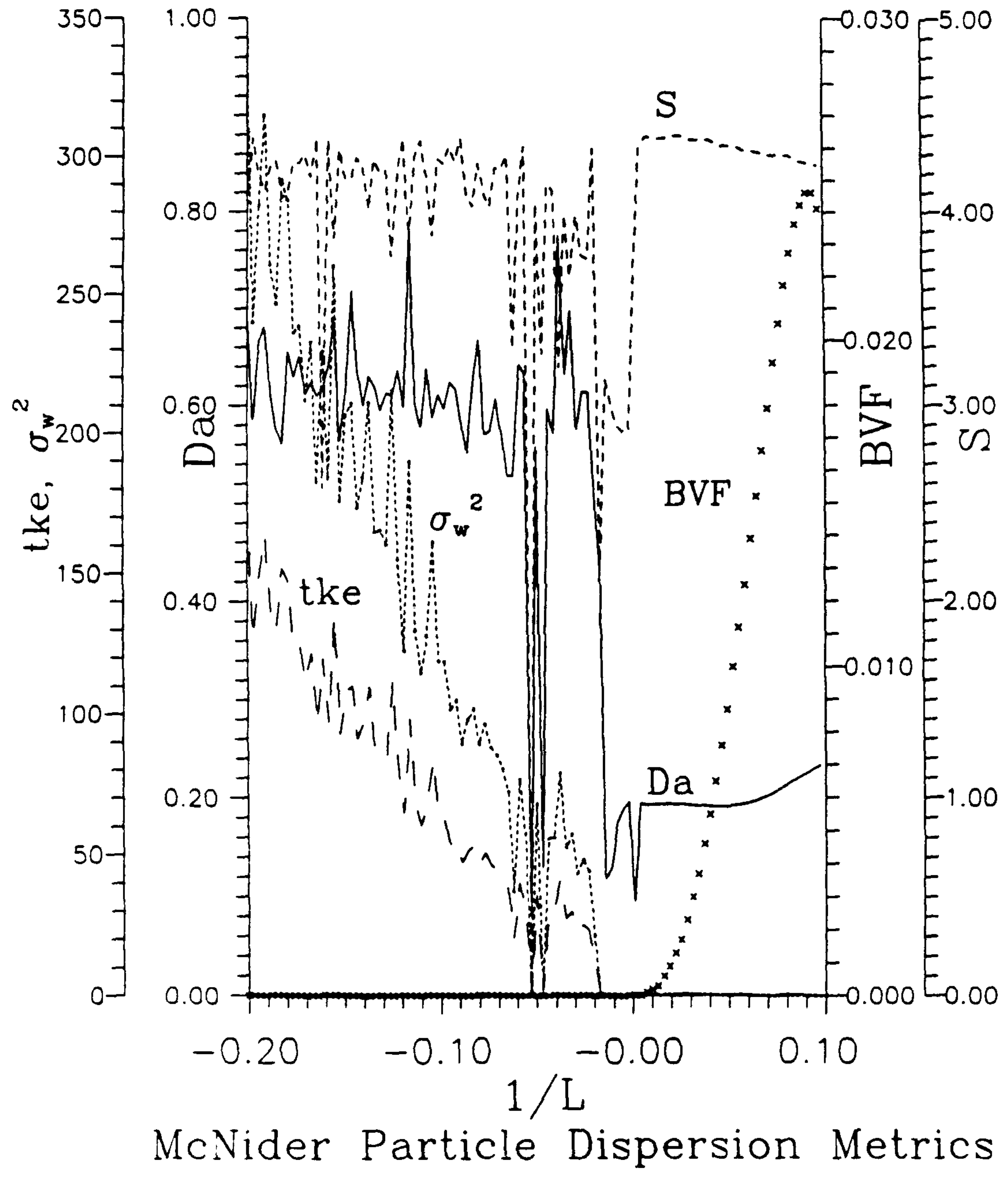

Figure 48. McNider Particle Dispersion Model. Test of phase velocity, with skewness.

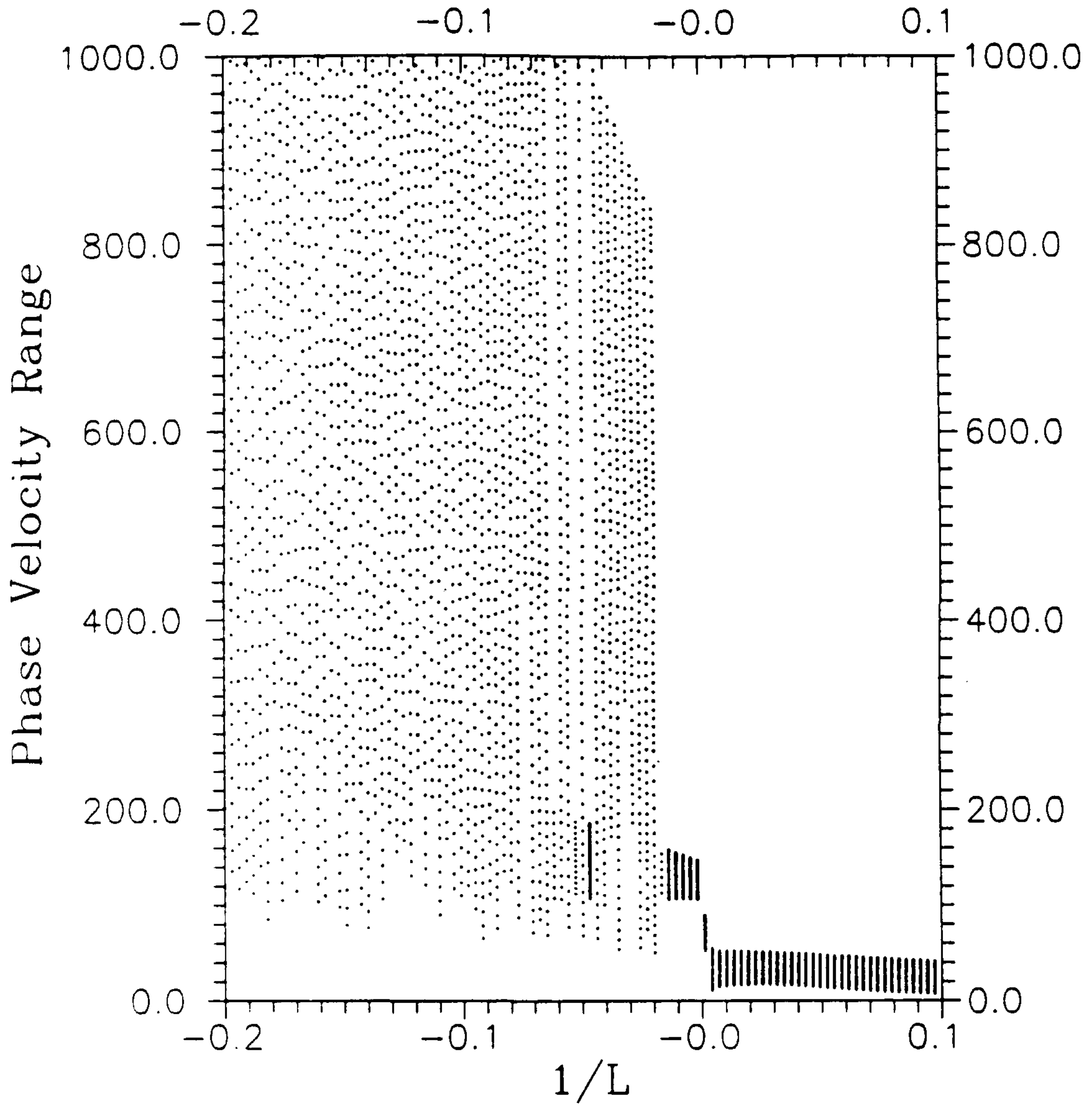

Figure 49. McNider Particle Dispersion Model Bifurcation Map. Test of phase velocity, with skewness.

more negative values of 1/L the σ_w^2/tke ratio indicates that nearly all the turbulence is in the vertical component which is not seen in the atmosphere nor is physically likely (Stull, 1988).

4. NPS Model, Skew On

The NPS model employed the double Gaussian skewness scheme as described in the theory section. It avoids the problems seen with the McNider model skewness scheme regarding the inappropriate high values for velocity, tke, and σ_w^2 for unstable values of 1/L. Also, the tke and σ_w^2 better reflect the behavior of real fluids: σ_w^2/tke exceeds 2/3 for unstable values of 1/L, but not for stable values. The sharp discontinuity in transition from unstable to stable 1/L is reduced somewhat.

Curiously, there is a distinct pattern to the strong fluctuations in D_A, S, tke, and σ_w^2 for unstable 1/L but not for stable 1/L values. This was also observed in the McNider skewness routine. The cause has not been determined; it could be due to limitations in the random number generator or perhaps an undiscovered program error. This is most obvious in the bifurcation map of vertical position range, Δz [Figure 53], where there are sharp jumps.

The phase velocity plots are similar to the total position plots in all cases examined. The reason is readily determined: the phase velocity is dominated by the total position, which varies from near zero to 1000 meters, while the velocity varies from near zero to roughly 3 meters/second.

As in the McNider plots there is little indication that the entropy is affected by a rising BVF, while D_A seems to rise as BVF rises, most notably in the vertical position plots. The BVFs indicate a long period compared with the 1/6 second timesteps. The entropy analysis will not detect a particle riding on a low-frequency wave with long period. In such a case the particle distribution over time may appear to be roughly uniform between a minimum and maximum value. This constraint is similar to the resolution requirements for entropy computation noted earlier. If the Δt in the model were increased from 1/6 second to say 10 seconds, while still retaining 3600 iterations, the entropy might also show a variance with the BVF.

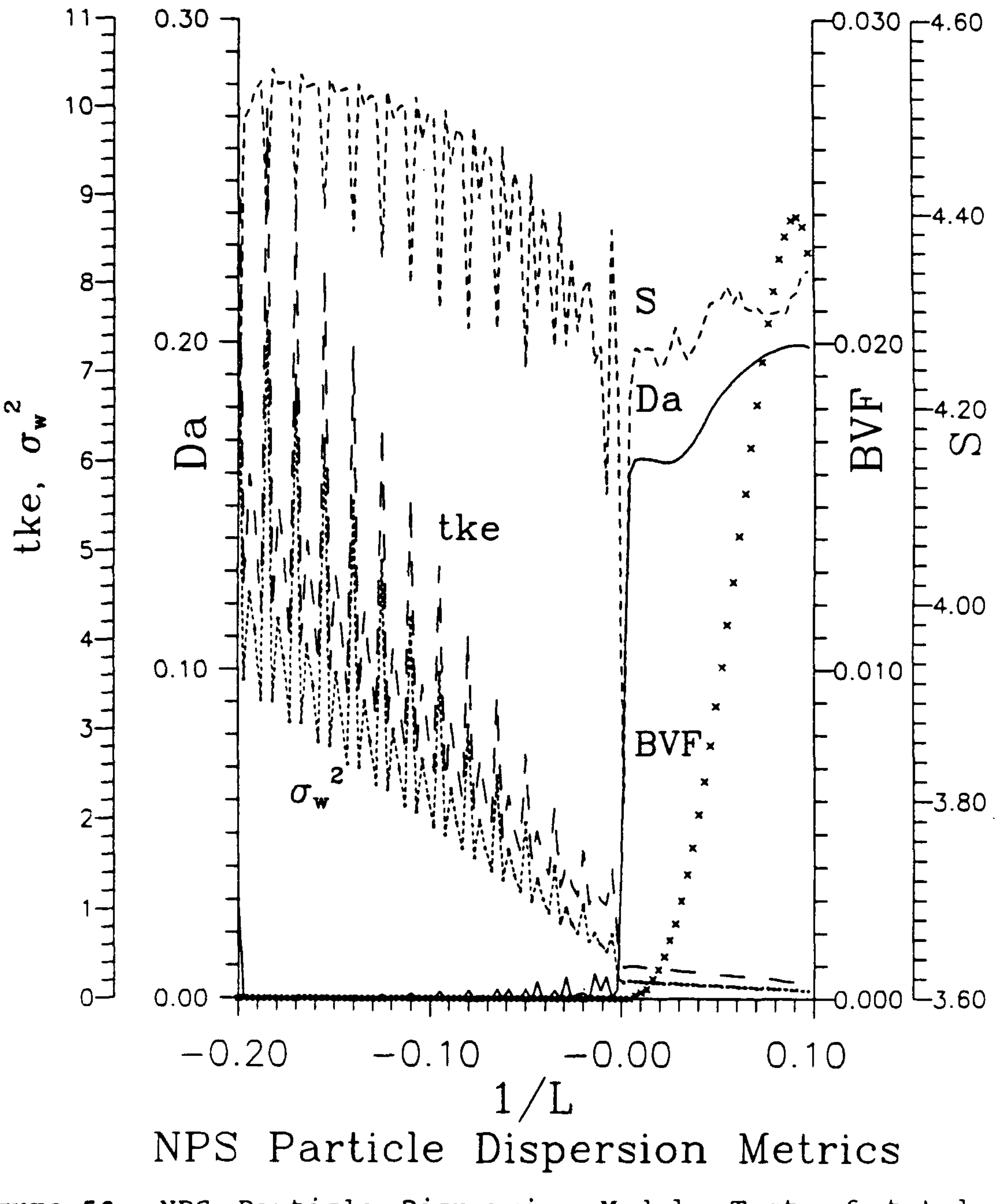

Figure 50. NPS Particle Dispersion Model. Test of total position, Δr, with skewness.

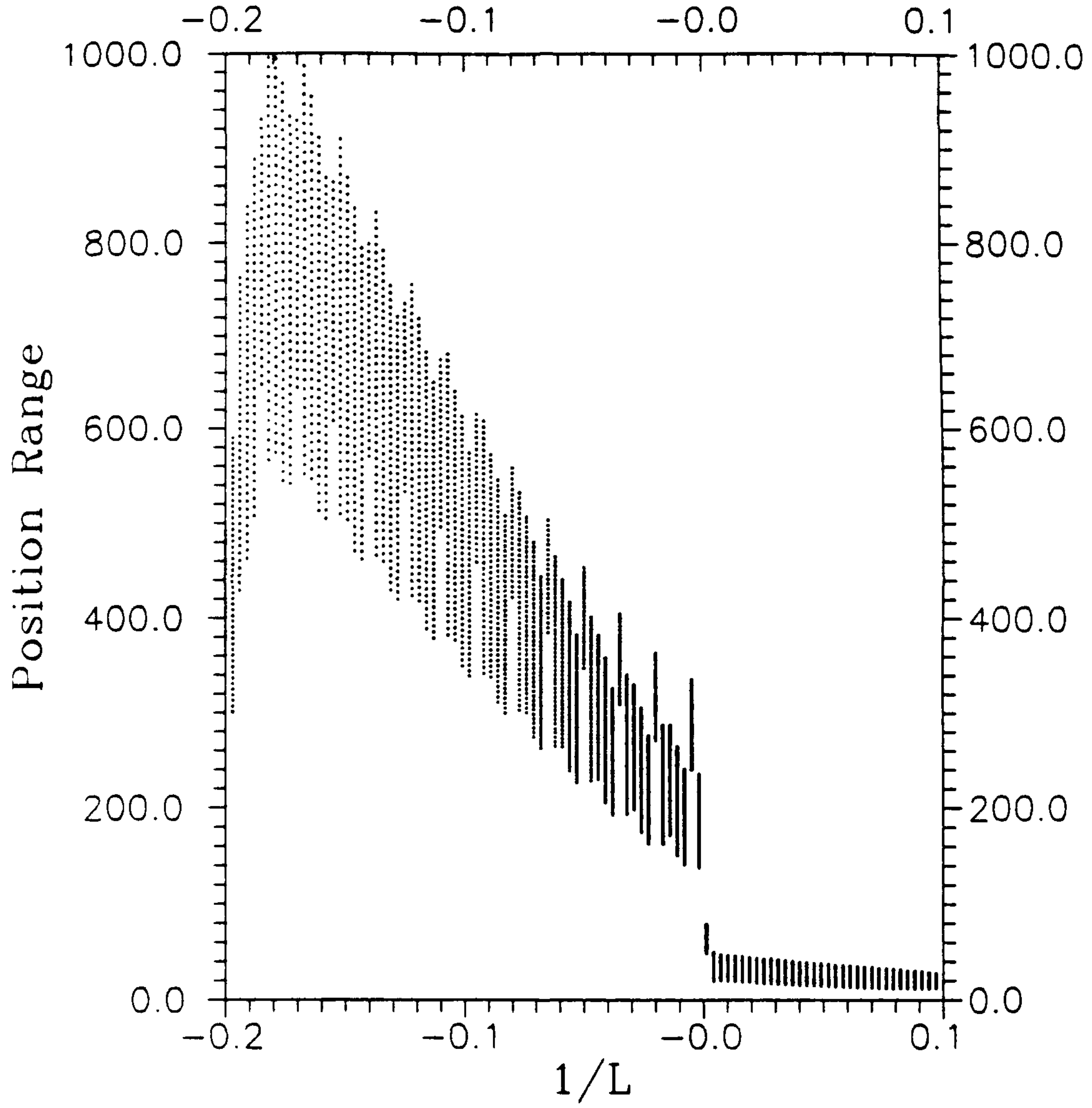

Figure 51. NPS Particle Dispersion Model Bifurcation Map. Test of total position, Δr, with skewness.

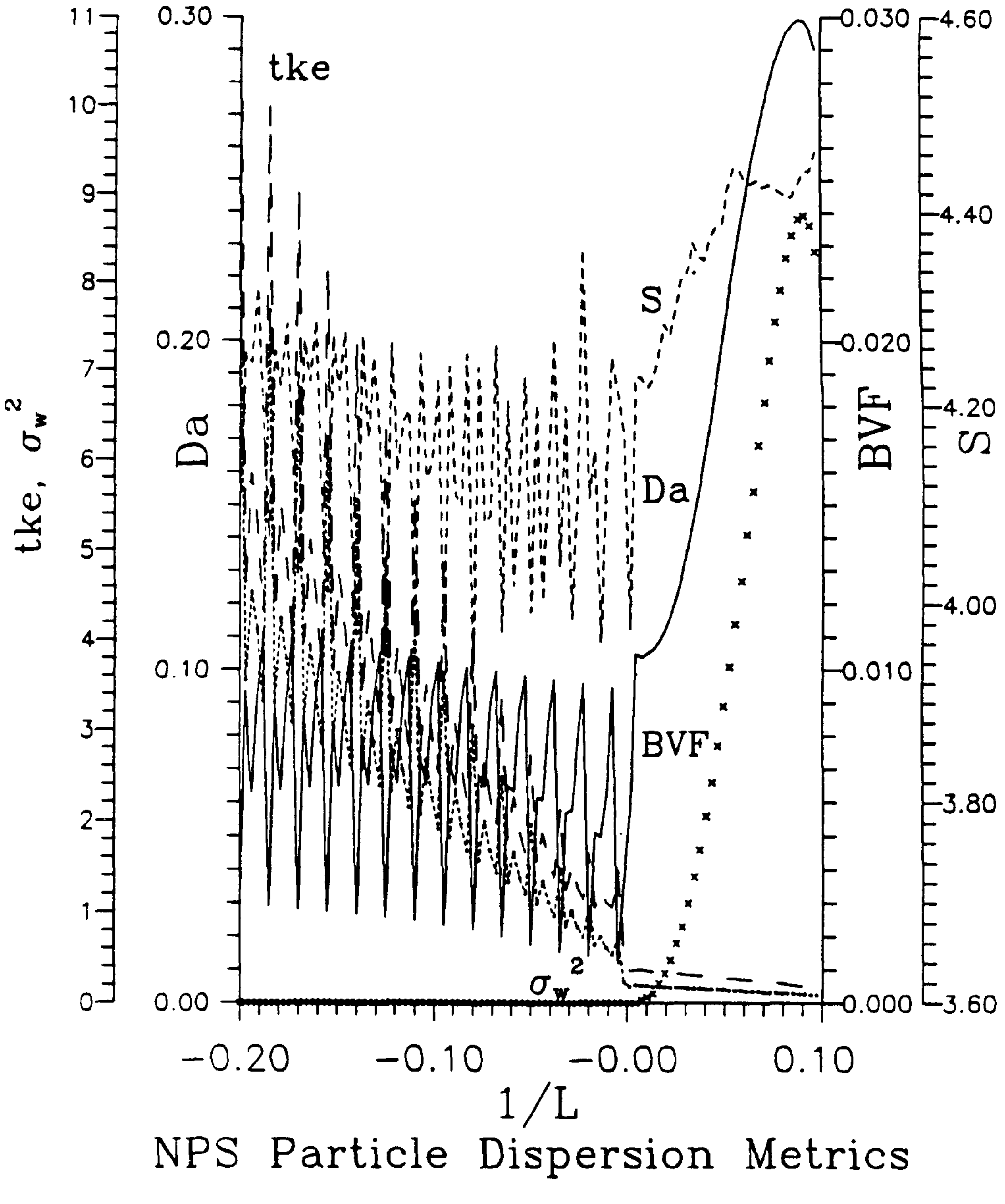

Figure 52. NPS Particle Dispersion Model. Test of vertical position, Δz, with skewness.

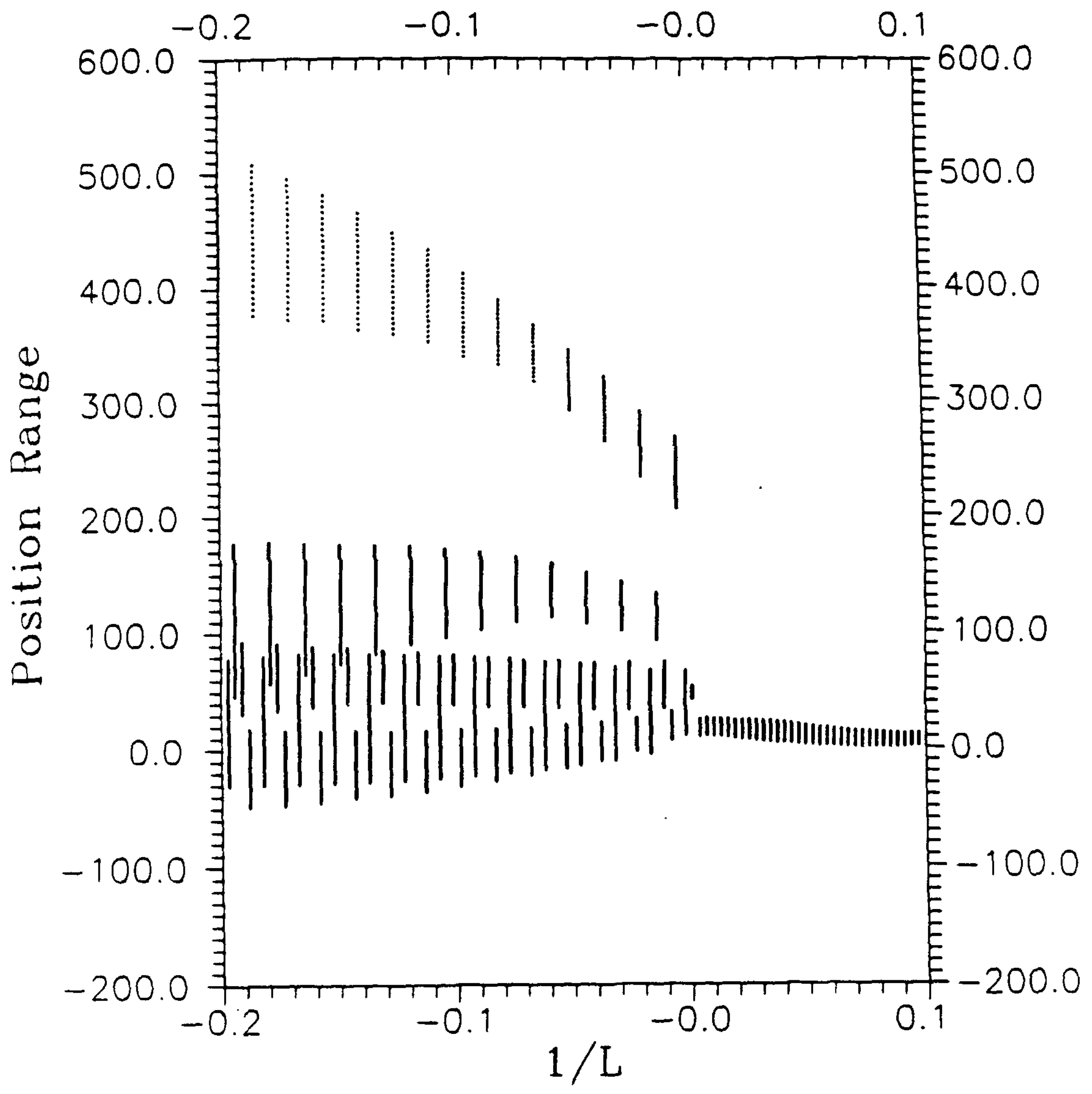

Figure 53. NPS Particle Dispersion Model Dispersion Map. Test of vertical position, Δz, with skewness.

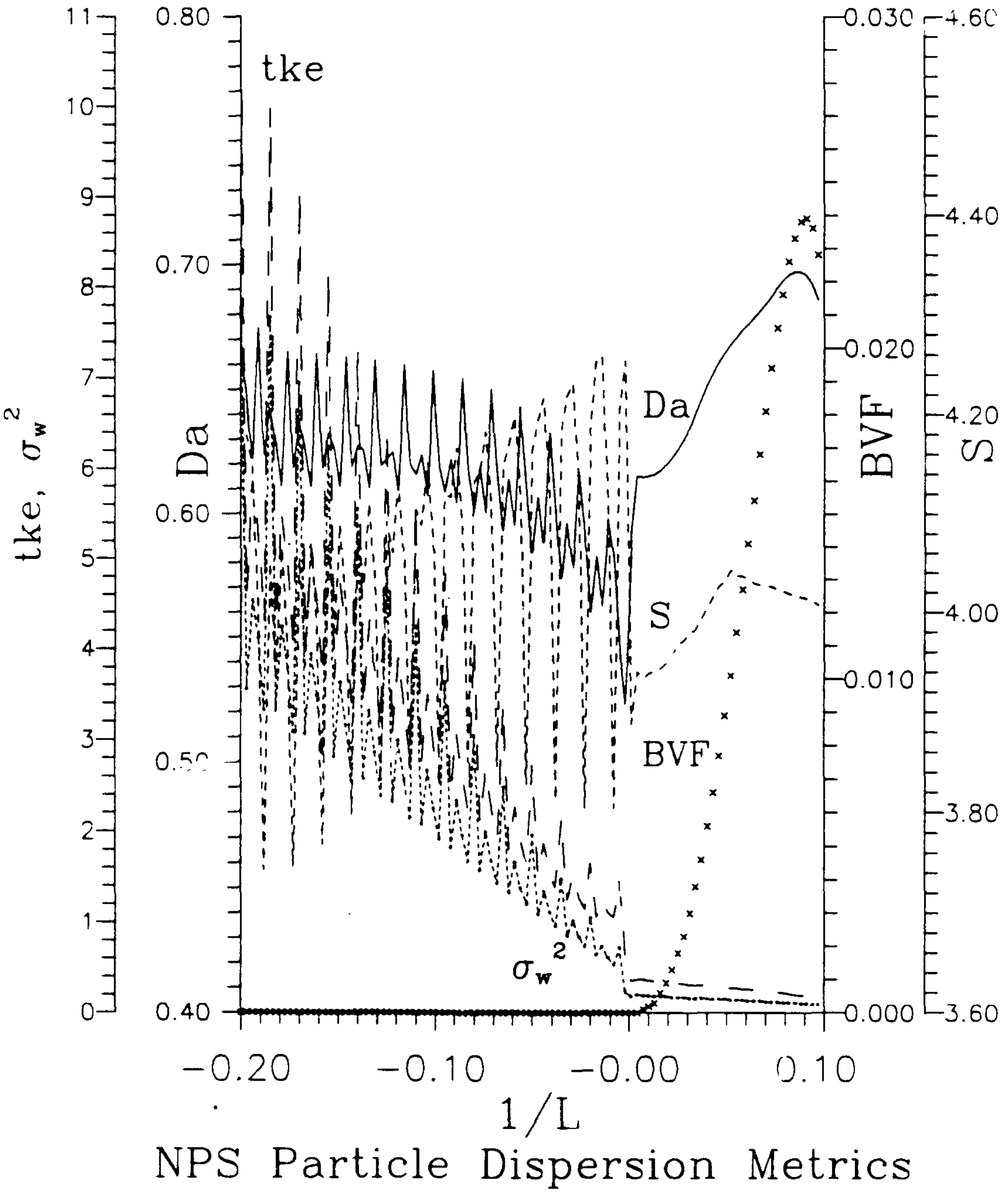

Figure 54. NPS Particle Dispersion Model. Test of velocity, with skewness.

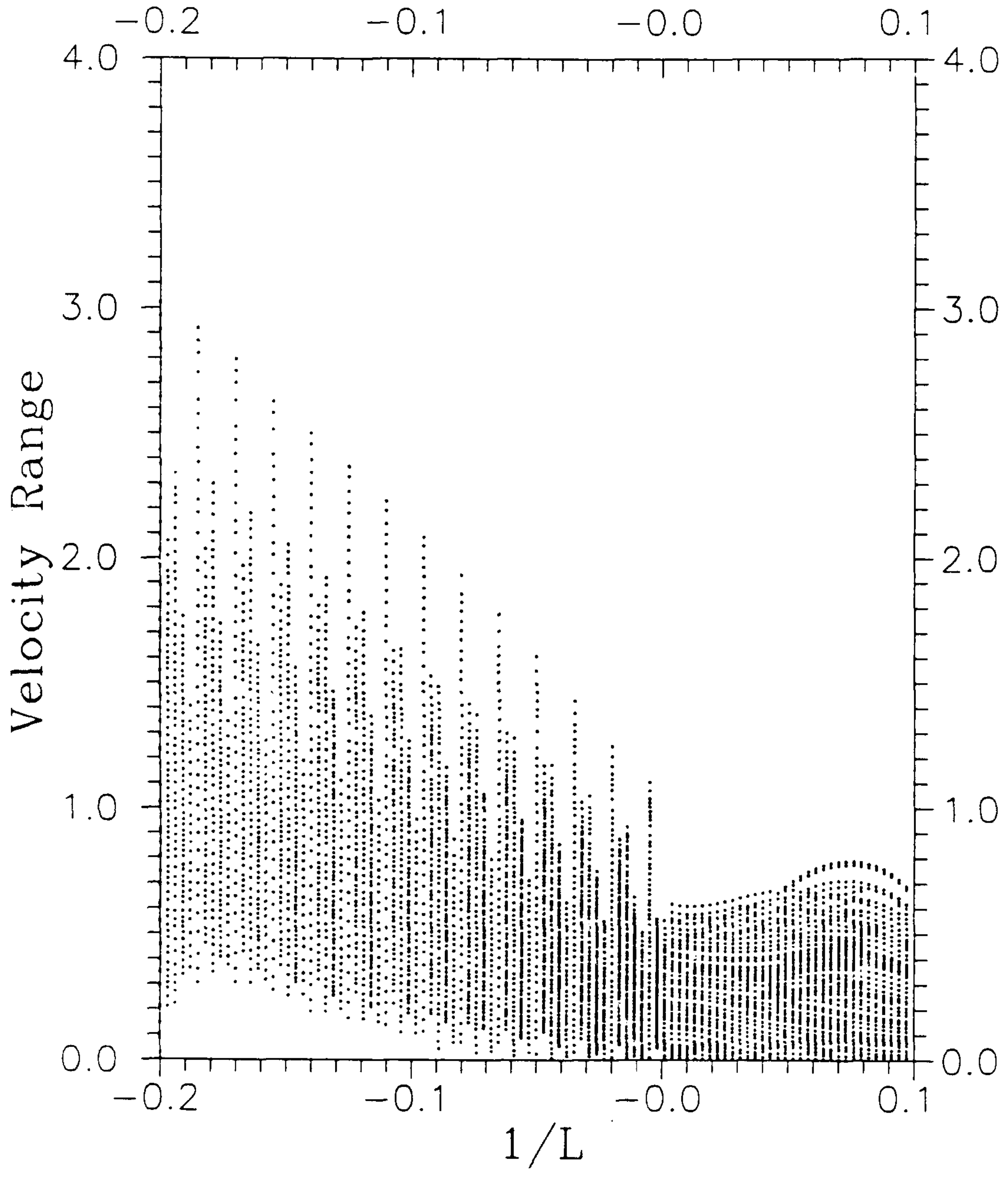

Figure 55. NPS Particle Dispersion Model Bifurcation Map. Test of velocity, with skewness.

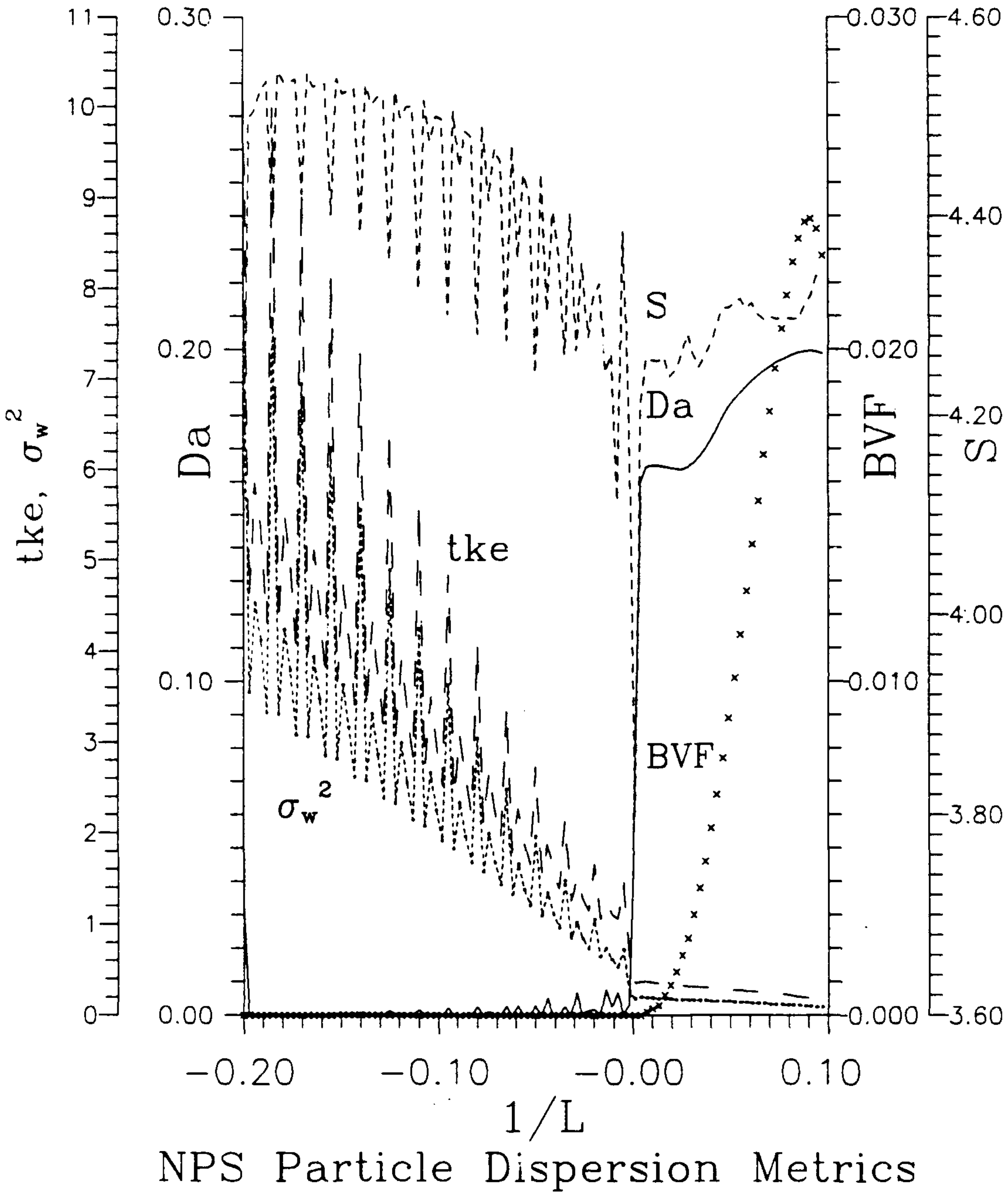

Figure 56. NPS Particle Dispersion Model. Test of phase velocity, with skewness.

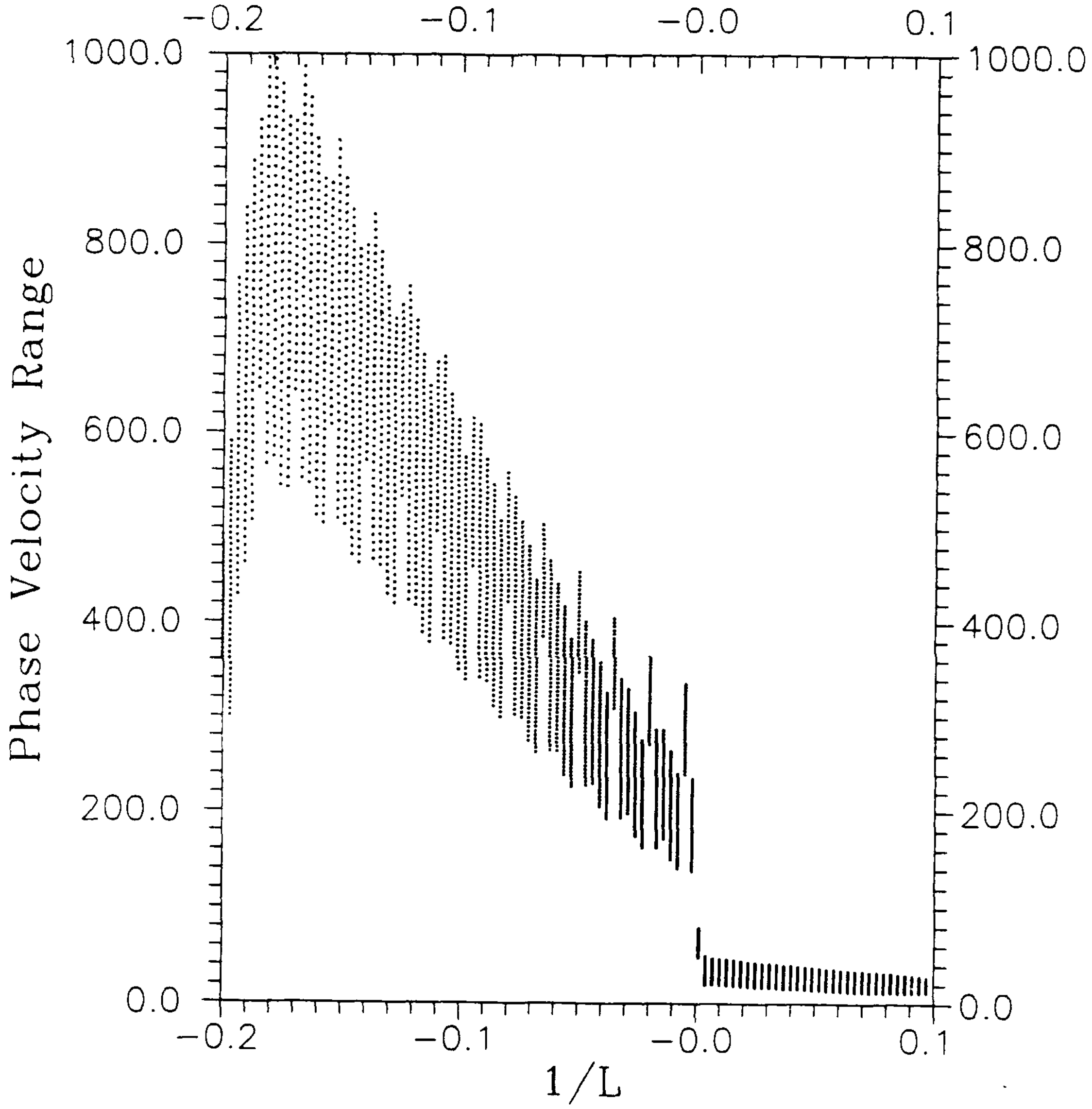

Figure 57. NPS Particle Dispersion Model Bifurcation Map. Test of phase velocity, with skewness.

IV. CONCLUSIONS

Chaos metrics such as the self-affine fractal dimension, D_A, entropy, S, and Lyapunov exponent, λ, are useful tools in the analysis of system characteristics and behavior. D_A measures the jaggedness of a trajectory, or relative magnitude of small versus large scale fluctuations. The entropy measures the randomness of state distribution. The Lyapunov exponent, difficult to implement in higher dimensions, measures the divergence of trajectories, and leads to a predictability time scale. As in both the chaos and atmospheric systems, D_A and S often correspond closely, but not always. The Henon equations also demonstrated that $\exp(D_A)$ is often of the same order as S.

These chaos metrics when used in conjunction with standard geophysical measures such as turbulent kinetic energy, tke, vertical velocity variance, σ_w^2, and the Brunt-Vaisala Frequency, BVF, can be applied usefully to atmospheric 3-D particle dispersion models. The McNider model as listed in Pielke (1984) has an inherent weakness in the skewness of vertical velocity variance. This weakness leads to obviously incorrect values of tke, σ_w^2, positions, and velocities for unstable values of the inverse Monin-Obukhov length, 1/L. The McNider algorithms also display an inherent discontinuity as the inverse Monin-Obukhov length crosses the zero point, reflecting a transition from unstable to stable buoyancy. This sharp discontinuity does not correspond to the real atmosphere, which suggests that model performance suffers at small values of 1/L.

The NPS particle dispersion model's measured performance seems to better reflect reality. it does not share the McNider model's weakness in the skewness of vertical velocity variance, and displays more realistic ratios of σ_w^2 to tke. The NPS Particle Dispersion Model also reduces the discontinuity in transition from negative to positive values of 1/L. However, the NPS model still has a problem with unstable inverse Monin-Obukhov lengths ($1/L < 0$); it displays a pattern to the vertical position distribution not reflective of real fluid behavior. Since the NPS model was developed in response to the current study, there may still be problems in coding or in the random number generator. The NPS model requires further refinement to model particle behavior in a fluid realistically.

Previous performance metrics for atmospheric particle models were only designed to measure aggregate particle behavior. The above chaos metrics parameters also offer some insight into the model behavior of individual particles. Results indicate that atmospheric dispersion models require further development at a fundamental level in replicating turbulent diffusion. For example, large fractal dimensions in atmospheric Lagrangian particle models seem indicative of strongly stable, rather than unstable, conditions, contrary to what seems likely for real turbulence. This points to the neglect in such models of motions at scales other than the dominant scale determined by $\sigma_{u,v,w}$. Thus, atmospheric Lagrangian particle models may simulate only single scale or at most a limited range of large scale diffusion processes. Fluctuations more in accord with real velocity spectra might display more realistic entropy and fractal behavior. As is, S and D_A behave oppositely at times, which indicates that diffusion and dispersion rates do not have equivalent meaning, even in the absence of mean windflow.

Simple periodic behavior preliminary to chaos in classical bifurcatory systems is also apparently not entirely equivalent to laminar wave behavior prior to the onset of turbulence. For example, if the time resolution of the analysis is much shorter than the period, the Shannon entropy for an atmospheric Laqrangian particle model may be quite large for wave motion, but small for periodic behavior. This is because the number of possible states in wave motion is not limited to the amplitude extrema as in the periodic motion.

One weakness of this study of Lagrangian particle model performance is the unavailability of real data for comparison purposes. As stated earlier, atmospheric data is normally obtained in the Eulerian rather than Lagrangian frame. This suggests a need for development of Lagrangian-based sensing in fluid turbulence experiments. Position and velocity measurements of particle markers in a wide range of atmospheric conditions, both stable and unstable, as a function of time, with rapid response remote sensors, would be useful in developing better models of particle dispersion and diffusion. As yet, since the gathering of data from real fluids in the Lagrangian frame is not feasible, these study results suggest that more extended application of chaos metrics to Eulerian measurements of turbulence in real fluids is warranted in order to gauge the performance of Eulerian turbulence models. Such applications need not be restricted to small-scale phenomena, since mesoscale Rayleigh-Benard convection also displays transitions from laminar cellular to chaotic behavior (Agee *et al.,* 1973). Model improvements based on such tests may then be extended to the Lagrangian frame.

With regard to the Eulerian frame, both fractal dimension and Shannon entropy are simple calculations which can be performed real-time *in situ* with current sampling devices, and should be simpler to implement than FFTs, since there are no transform matrices. One area to consider when conducting measurements of the self-affine fractal dimension is the size of both ϵ, the time increment, and T, the time window width over which the length is being defined. ϵ has a minimum size dictated by the response time of the sensors, while the size of T is critical when looking for intermittent or low frequency events, e.g., gravity waves governed by the Brunt-Vaisala Frequency. If ϵ_{min} is too small, then the time increment will fall in the range where the velocity autocorrelation decay is not exponential but constrained to be parabolic due to continuity and the viscosity of real fluids. At the same time ϵ_{min} must be at least ~ 3 orders of magnitude smaller than T to establish statistical validity. Establishing the appropriate ϵ_{min} is crucial to distinguish waves from turbulence, and probably applies equally for distinguishing intermittent and coherent structures in general. (Kamada and DeCaria, 1992)

Another area for further study is using the Lyapunov exponent in analyzing 3-D turbulence. Although difficult to compute in 3-D, λ provides a definite test of chaos in the particle diffusion rate. A study of all three chaos metrics seems necessary to clarify the relationship of diffusion rate to dispersion rate in these models.

Standard geophysical turbulence measures such as tke and σ_w^2 might also be applied to classical chaotic systems such as the logistics difference and Henon equations. The definitions of timescale and averaging time must first be established, e.g., one iteration equals one time unit. The corresponding tke would be zero for a fixed point, and definite magnitude changes would occur as the function bifurcates to periods 2, 4, 8, and to chaos.

The utility of chaos metrics in analyzing the 3-D Monte Carlo based particle dispersion models also suggests possible utility for other laminar/turbulent phenomena. These might include plasmas, acoustics, and laser cavities. Solid state free electron gas systems may also find these chaos metrics useful in describing phonon and electron transport. Phonon resonance in crystal lattices also reminds one that particle behavior in lattice gases and cellular automata may be studied with such chaos metrics. These measures might also be useful in modeling gas or liquid phase complex chemical kinetics or radiation, which have long been simulated by Monte Carlo schemes.

APPENDIX A. LOGISTICS DIFFERENCE EQUATION ANALYSIS PROGRAM

```
C*************************************************************************
    PROGRAM Feigenbm
c
c  This program computes self-affine fractal dimension, Da, for the
c  Feigenbaum recursion relation,
c
c       x(i+1) = 4 lambda x(i)( 1 - x(i),
c
c
c  as a function of lambda.  Da is defined for the iteration series as:
c
c                 d Ln[ L(i) ]                           /T
c       Da = - ---------------- , where  L(i) = (1/i)|   |F(i+1) - F(i)|di
c                 d Ln[ i ]                              Jo
c
c                                                    N
c                                              =  SUM |F(i) - F(i-1)|
c                                                 i=k,k
c
c  and k is an exact integer divisor of N.  Here we choose
c  N = 3,600 to give us at least three decades to use in computing Da.
c  This also allows 48 exact integer divisors of N in the k array.  So
c  this gives us 48 points of i vs. L(i) in the linear regression for Da.
C*************************************************************************
c
      DOUBLE PRECISION lambda, lglngth(48, 100), lgk(48), y, sum, x
      DIMENSION y(0:3600), sum(48, 100), k(48), nyval(48), xy(2,48)
     &          ans(16), Dalmbda(3,100)
      DATA k/1, 2, 3, 4, 5, 6, 8, 9, 10, 12, 15, 16, 18, 20, 24, 25,
     &     30, 36, 40, 45, 50, 60, 72, 75, 80, 90, 100, 103, 106, 109,
     &    116, 120, 144, 150, 180, 200, 225, 240, 300, 360, 400, 450,
     &    600, 720, 900, 1200, 1800, 3600/
      OPEN (1, FILE='feigenis.dat', RECL=255, STATUS='new', ERR=700)
      OPEN (2, FILE='feigenll.dat', RECL=255, STATUS='new', ERR=800)
c
      y(0) = 0.0
c Get Da for different values of lambda from 0 to 1.
      m = 100
      DO 500 l = 1, m
         el = l
         lambda = 0.01*el
         IF (lambda .gt. 0.9999) lambda = 0.9999
         x      = 0.01
c Create the iteration series for a given lambda
         N      = 7200
         WRITE (1, *)'   i      y(i)   '
         DO 200 i = 1, N
            y(i) = 4.*lambda*x*(1-x)
            x    = y(i)
            WRITE (1,150) i, y(i)
150         FORMAT (1x, I4, 3x, f8.5)
200         CONTINUE
c With iteration series, get lengths and self-affine fractal dimension
         WRITE (2, *)'   Ln(ε)   Ln [L(ε)]'
```

```
c Get total length as sum of differences for different spacings of i
         DO 400 j = 1, 48
            sum(j,l) = 0.
            DO 300 i = 3600 + k(j), 7200, k(j)
               sum(j,l) = sum(j,l) + abs( y(i) - y( i-k(j) ) )
d              IF (j .gt. 44)
d    &            WRITE(6,250) y(i),y(i-k(j)),sum(j,l),i,j,k(j)
d250              FORMAT ( 1x, 3f8.4,3I5)
300            CONTINUE
d           IF (j .gt. 44)
d    &      WRITE(6,*)'sum(',j,',l) = ',sum(j,l)
            lglngth(j,l) = alog10( sum(j,l) )
            lgk(j) = alog10( float(k(j)) )
            xy(1,j) = lgk'+`
            xy(2,j) = lglngth(j,l)
            WRITE (2,350) lgk(j), lglngth(j,l)
350         FORMAT (1x, f8.5, 3x, f8.5)
400         CONTINUE
c
c  Do linear regression to get Da from log10(k) vs. log10 [L(ε)] values
c
         CALL STLNRG(n, nyval, xy, ans)
c
c  -ans(2) contains Da
c  ans(14) contains the standard deviation of Da
c
c  Put lambda, Da, and σDa in a 3 by 1 array
c
         Dalmbda(1,l) = -ans(2)
         Dalmbda(2,l) = -ans(14)
         Dalmbda(3,l) = lambda
500      CONTINUE
      WRITE (3,550) ((Dalmbda(i,l), i = 1,3), l = 1,m)
550   FORMAT (1x, 3(f9.5,2x))
      STOP
700   WRITE (6,*)'error opening feigenis.dat'
      STOP
800   WRITE (6,*)'error opening feigenll.dat'
      STOP
      END
c**********************************************************************
    subroutine stlnrg (k, nyval, xy, ans)
c
c  This subroutine computes the slope and other statistics for a
c  linear regression with several y values for each x value or with
c  one independent variable.
c
c  argument   use     description
c     k       input   Number of different x values
c
c     nyval   input   Array of number of y values for each x
c                     value. Dimensioned k
c     xy      input   array of x and y pairs (x1, y1, x2, y2, etc.)
c                     of dimension (2,k)
c
c     ans     output  Array of results computed, dimensioned 16.
c
c                     ans
c                     1 Number
c                    *2 Slope
c                     3 Mean of y
```

```
c                       4 Mean of x
c                       5 y-intercept
c                       6 Sum of y**2
c                       7 Sum of squares mean
c                       8 Sum of squares slope
c                       9 Residual
c                      10 standard deviation of x
c                      11 standard deviation of y
c                      12 standard deviation error
c                      13 standard deviation y-bar
c                     *14 standard deviation slope
c                      15 standard deviation y-intercept
c                      16 F-Ratio for slope
c
c***********************************************************************
    DIMENSION nyval(1), xy(1), ans(16)
      sumx = 0.0
      sumy = 0.0
      sumx2 = 0.0
      ans(6) = 0.0
      sumxy = 0.0
      n = 0
      nx = 1
      ny = 2
      DO 20 j = 1,k
      nyval(j) = 1
      m = nyval(j)
      n = n + m
      DO 10 i = 1, m
      sumx = sumx + xy(nx)
      sumx2 = sumx2 + xy(nx)*xy(nx)
      sumy = sumy + xy(ny)
      ans(6) = ans(6) + xy(ny)*xy(ny)
      sumxy = sumxy + xy(nx)*xy(ny)
10    ny = ny + 1
      nx = nx + nyval(j) + 1
20    ny = ny + 1
      en = n
      ans(1) = en
      s1 = ans(1)*sumx2 - sumx*sumx
      s2 = ans(1)*sumxy - sumx*sumy
      en1 = en - 1.0
      en2 = en1 - 1.0
      ans(2) = s2/s1
      ans(3) = sumy/en
      ans(4) = sumx/en
      ans(5) = ans(3) - ans(2)*ans(4)
      ans(7) = ans(3)*sumy
      ans(8) = ans(2)*s2/en
      ans(9) = ans(6) - ans(7) - ans(8)
      s4 = ans(9)/(en - 2.0)
      endsl = en/s1
      ans(10) = sqrt(1.0/(en1*endsl))
      ans(11) = sqrt((ans(6) - ans(7))/en1)
      ans(12) = sqrt(s4)
      a13 = s4/en
      ans(13) = sqrt(a13)
      a14 = s4*endsl
      ans(14) = sqrt(a14)
      ans(15) = sqrt(a13 + a14*ans(4)*ans(4) )
      ans(16) = ans(8)/s4
      RETURN
      END
C***********************************************************************
```

APPENDIX B. HENON EQUATION ANALYSIS PROGRAM

```
C *****************************************************************
      PROGRAM CHAOS
C
C  Written by Ray Kamada & Korey Jackson
C
      DOUBLE PRECISION X(5000), Y(5000), DA, A, B,
     +  DAERR, L(100), B1, S
      INTEGER NMAX,I1,JMAX,IFLAG, J1, NMAX1, COUNT, EPSIL(100)
      OPEN(2, FILE='LENGTH.DAT', STATUS='NEW', ERR=150)
      WRITE(2,*)'   A     B          DA                    DAERR'
      OPEN(7, FILE='ENTROP.DAT', STATUS='NEW')
      WRITE(7,*)'   A     B          S'
C
C Establish NMAX
C
      NMAX=3600+1
      NMAX1=NMAX+100
      JMAX=NMAX/2
      PRINT*,'JMAX=',JMAX
      IF (NMAX.GT.5000) GO TO 100
C Initialize all variables
       IFLAG=0
       CALL DIVISR(NMAX, JMAX, EPSIL, COUNT)
C
C Now set up a chaos generating routine...
C
      A=1.0570D0
      B1=0.3D0
       I1=0
C30    IF(I1.GT.600)GOTO 55
          J1=0
C         A=A+0.0010D0
          B=B1
          I1=I1 + 1
c 35        IF(J1.GT.30)GOTO 50
c            J1=J1+1
c            B=B+0.05D0
        print*,'MAIN1  A, B,',A,B
           IFLAG=0
      CALL HENON(NMAX1,A,B,X,Y,IFLAG)
      print*,'henon complete',a,b,iflag
C  Error trap for x, y out of range (diverging solutions)
c       IF(IFLAG.EQ.1)GOTO 35
C Comment out above line & replace w/below line when B is fixed
C      IF(IFLAG.EQ.1)GOTO 30
      if(iflag.eq.1.)goto 200
C
C  Analyze the data
C
      CALL ANALYS(NMAX,JMAX,X,Y,A,B, DA, DAERR, COUNT, EPSIL)
      print*,'analysis complete', a, b, DA
44    WRITE(2,45) A, B, DA, DAERR
45    FORMAT( F7.4, 3X, F6.3, 5X, F8.5, 5X, F8.5)
46    print*,'WRITE to file complete in analysis'
      CALL ENTROP(NMAX,X,Y,S)
      WRITE(7,47) A, B, S
47    FORMAT(F7.4,3X, F6.3,5X,F8.5)
c 49    GOTO 35
50    CONTINUE
C 54    GOTO 30
C
55     CONTINUE
C      CLOSE(2)
C      CLOSE(7)
      GOTO 200
C
100   PRINT*,'ERROR....NMAX Greater than 10000....Array size too small'
      GOTO 200
146   PRINT*,'ERROR...Can not open henon.dat file'
      GOTO 200
148   PRINT*,'ERROR...Can not open analys.dat file'
      GOTO 200
150   PRINT*,'ERROR...Can not open length.dat file'
200   PRINT*,'COMPLETED MAIN2'
      END
C ******************************************************************
      SUBROUTINE DIVISR(NMAX,JMAX,EPSIL, COUNT)
      INTEGER CHECK, CHECK1, COUNT, NMAX, JMAX, EPSIL(100)
      CHECK=0
      CHECK1=0
      COUNT=0
      NMAX=NMAX-1
1000  CHECK=CHECK+1
      IF(CHECK.GT.JMAX) GOTO 1050
      CHECK1=NMAX/CHECK
      CHECK1=CHECK1*CHECK
      IF(CHECK1.NE.NMAX) GOTO 1000
      COUNT=COUNT+1
      EPSIL(COUNT)=CHECK
      PRINT*,'EPSIL=',EPSIL(COUNT), COUNT
      GOTO 1000
1050  CONTINUE
      NMAX=NMAX+1
      RETURN
      END
C ******************************************************************
      SUBROUTINE HENON(NMAX1,A,B,X,Y,IFLAG)
      DOUBLE PRECISION X(NMAX1), Y(NMAX1), A, B, CHECK
      INTEGER NMAX1, N, IFLAG, NMAX
C     OPEN(4,FILE='HENON.DAT', STATUS='NEW', ERR=700)
C
C  Initialize variables
C
      PRINT*,'PASSED TO HENON OK'
      DO 300 N=1,NMAX1
        X(N)=0.0D0
        Y(N)=0.0D0
300   CONTINUE
C
C MAIN COMPUTATION ROUTINE
C
C  First initiate X(1), if desired.
       X(1)=-0.65d0
       Y(1)= 0.38d0
C
c     WRITE(4 *)'     N           X            Y'
```

```
      DO 400 N=1, NMAX1
        X(N+1)=1.-A*X(N)*X(N)+Y(N)
        CHECK=X(N+1)
        IF(CHECK.GE.100.OR.CHECK.LE.-100.) THEN
           IFLAG=1
           PRINT*,'PROGRAM OUT OF RANGE IN X'
        ENDIF
        Y(N+1)=B*X(N)
        CHECK=Y(N+1)
        IF(CHECK.GE.1000.OR.CHECK.LE.-1000.0) THEN
           IFLAG=1
           PRINT*,'PROGRAM OUT OF RANGE IN Y'
        ENDIF
        IF(IFLAG.EQ.1)GOTO 760
         IF(N.LT.101)GOTO400
C     WRITE(4,350) N, X(N), Y(N)
350   FORMAT(I10,5X,E15.5,5X,E15.5)
400   CONTINUE
C
C  NOW SHIFT THE VALUES DOWN IN X(Y), Y(I)
C
      NMAX=NMAX1-100
      DO 600 N=1, NMAX
         X(N)=X(N+100)
         Y(N)=Y(N+100)
600    CONTINUE
C
C Return to the main program
C
      PRINT*,'COMPLETED HENON OK FOR',A,B
      GO TO 770
C
700   PRINT*,'Error in opening HENON.DAT file'
C 740   CLOSE(2)
C 750   CONTINUE
       GOTO 770
760   PRINT*,'X value out of range...A,B TOSSED', A, B
      IFLAG=1
770   CLOSE(1)
      RETURN
      END
C ****************************************************************
      SUBROUTINE ANALYS(NMAX,JMAX, X,Y,A,B, DA, DAERR, COUNT,
     + EPSIL)
      DOUBLE PRECISION SCRAP1, SCRAP2, L(100), X(NMAX), SUM(100),
     +  Y(NMAX), EPS(100), DA, A, B, DAERR, SCRAP3, XY(200),
     +  ANS(16)
       INTEGER NMAX, I, J, CHECK, CHECK1,JMAX,COUNT, EPSIL(100),
     +  COUNT2, nyval(100), COUNT3
C
C   VARIABLE LIST:
C                  NMAX=MAXIMUM NUMBER OF DATA POINTS
C                  COUNT=NUMBER OF EVEN DATA POINTS THAT WILL
C                          DIVIDE EVENLY INTO NMAX
C                  L(J)=LENGTH FOR A GIVEN EPSILON
C                  EPSIL(J)=WIDTH, OR AN INTEGER DIVISOR OF NMAX
C                  EPS(J)=EPSIL(J) IN INTEGER FORM
C                  DAERR=STANDARD DEVIATION OF DA (LOG FORM)
C
      OPEN (3, FILE='ANALYS.DAT', STATUS='NEW',
     +         ERR=9010)
```

```
C
C     INITITALIZE THE VARIABLES
C
      DO 4000 J=1, 100
         L(J)=0.0D0
         SUM(J)=0.0D0
4000   CONTINUE
C
C     Now compute L(epsil)
C
      DO 7500 J=1, COUNT
      CHECK=NMAX-EPSIL(J)
      CHECK1=EPSIL(J)
       DO 7000 I=CHECK1, CHECK, CHECK1
         SCRAP1=X(I+1)-X(I-CHECK1+1)
         SCRAP2=Y(I+1)-Y(I-CHECK1+1)
         SCRAP1=SCRAP1*SCRAP1
         SCRAP2=SCRAP2*SCRAP2
         SCRAP2=SCRAP2+SCRAP1
         SCRAP2=DSQRT(SCRAP2)
         SUM(J)=SUM(J)+SCRAP2
7000  CONTINUE
      L(J,=SUM(J)
7500  CONTINUE
C
C  QUIK DATA PRINT
      DO 7550 J=1, COUNT
         L(J)=L(J)+1.0D0
         SCRAP1=L(J)
         L(J)= DLOG10(SCRAP1)
         SCRAP3=DBLE(EPSIL(J))
         EPS(J)=DLOG10(SCRAP3)
       WRITE(3,7545) EPS(J), L(J)
7545  FORMAT(5X,F8.5,5X,F8.5)
7550  CONTINUE
C
C Now do a linear regression to find DA
C   First put into a format to use the canned linear regression
C     subroutine.
      COUNT3=2*COUNT
       DO 7700 J=2, COUNT3, 2
         I=J/2
         XY(J-1)=EPS(I)
         XY(J)=L(I)
7700  CONTINUE
C
c COUNT2 IS THE COUNT WITH THE LAST 3 VALUES TOSSED FOR Da
c   computation
      COUNT2=COUNT-3
      CALL LINREG (COUNT2, nyval, XY, ANS)
      print*,'made it out of linreg'
      DA=ANS(2)
      DA=-DA
      DAERR=ANS(14)
      GOTO 9100
9000  PRINT*,'ERROR OPENING HENON.DAT FILE'
9010  PRINT*,'ERROR OPENING ANALYS.DAT FILE'
9020  PRINT*,'ERROR OPENING LENGTH.DAT FILE'
9100  print*,'made it to 9100'
      RETURN
      END
```

```
C *****************************************************************
      subroutine LINREG (k, nyval, xy, ans)
c
c  This subroutine computes the slope and other statistics for a
c  linear regression with several y values for each x value or with
c  one independent variable.
c
c  argument   use     description
c     k       input   Number of different x values
c
c     nyval   input   Array of number of y values for each x
c                     value. Dimensioned k
c     xy      input   array of x and y pairs (x1, y1, x2, y2, etc.)
c                     of dimension (2,k)
c
c     ans     output  Array of results computed, dimensioned 16.
c
c                     ans
c                      1 Number
c                     *2 Slope
c                      3 Mean of y
c                      4 Mean of x
c                      5 y-intercept
c                      6 Sum of y**2
c                      7 Sum of squares mean
c                      8 Sum of squares slope
c                      9 Residual
c                     10 standard deviation of x
c                     11 standard deviation of y
c                     12 standard deviation error
c                     13 standard deviation y-bar
c                    *14 standard deviation slope
c                     15 standard deviation y-intercept
c                     16 F-Ratio for slope
c ****************************************************************
      DOUBLE PRECISION nyval(1), xy(1), ans(16), s1, s2, sumx, sumx2
     + sumy, sumy2, sumxy, en1, en2, s4, endsl, a13, a14
      sumx = 0.0D0
      sumy = 0.0D0
      sumx2 = 0.0D0
      ans(6) = 0.0D0
      sumxy = 0.0D0
      n = 0
      nx = 1
      ny = 2
      DO 20 j = 1,k
      nyval(j) = 1
      m = nyval(j)
      n = n + m
      DO 10 i = 1, m
      sumx = sumx + xy(nx)
      sumx2 = sumx2 + xy(nx)*xy(nx)
      sumy = sumy + xy(ny)
      ans(6) = ans(6) + xy(ny)*xy(ny)
      sumxy = sumxy + xy(nx)*xy(ny)
10    ny = ny + 1
      nx = nx + nyval(j) + 1
20    ny = ny + 1
      en = n
      ans(1) = en
      s1 = ans(1)*sumx2 - sumx*sumx
      s2 = ans(1)*sumxy - sumx*sumy
      en1 = en - 1.0
      en2 = en1 - 1.0
      ans(2) = s2/s1
      ans(3) = sumy/en
      ans(4) = sumx/en
      ans(5) = ans(3) - ans(2)*ans(4)
      ans(7) = ans(3)*sumy
      ans(8) = ans(2)*s2/en
      ans(9) = ans(6) - ans(7) - ans(8)
      s4 = ans(9)/(en - 2.0)
      endsl = en/s1
      ans(10) = sqrt(1.0/(en1*endsl))
      ans(11) = sqrt((ans(6) - ans(7))/en1)
      ans(12) = sqrt(s4)
      a13 = s4/en
      ans(13) = sqrt(a13)
      a14 = s4*endsl
      ans(14) = sqrt(a14)
      ans(15) = sqrt(a13 + a14*ans(4)*ans(4) )
      ans(16) = ans(8)/s4
      RETURN
      END
C ***************************************************************
C   23 Apr 92 revision (goes with 0<a<1.25 plot)
C
      SUBROUTINE ENTROP(NMAX,X,Y,S)
      DOUBLE PRECISION X(NMAX), Y(NMAX), S, XMAX, XMIN, YMAX, YMIN,
     +  PROB(200,200), INVNMA, P, XRANGE, YRANGE
      INTEGER IX, IY, I, J, NMAX
C  First find maximum and minimum values
      XMAX=0.0D0
      YMAX=0.0D0
      XMIN=0.0D0
      YMIN=0.0D0
      print*,'finding max and minimum values'
      DO 2000 I=1, NMAX
         IF(X(I).GT.XMAX) XMAX=X(I)
         IF(X(I).LT.XMIN) XMIN=X(I)
         IF(Y(I).GT.YMAX) YMAX=Y(I)
         IF(Y(I).LT.YMIN) YMIN=Y(I)
2000  CONTINUE
C  Now initialize the probability array
      print*,'initializing P array'
      DO 2105 I=1, 200
         DO 2100 J=1, 200
            PROB(I,J)=0.0D0
2100     CONTINUE
2105  CONTINUE
C Count the number of points in each unit area
C First set up the scheme
      XRANGE=XMAX-XMIN
      XRANGE=2.00D2/XRANGE
      YRANGE=YMAX-YMIN
      YRANGE=2.00D2/YRANGE
      DO 2110 I=1, NMAX
         IX=INT((X(I)-XMIN)*XRANGE)+1
         IY=INT((Y(I)-YMIN)*YRANGE)+1
         IF(IX.GT.200) IX=200
         IF(IY.GT.200) IY=200
         PROB(IX IY)=PROB(IX,IY)+1.0D0
```

```
2110  CONTINUE
C  Now normalize
      INVNMA=1.0D00/DBLE(NMAX)
      DO 2120 IX=1, 200
         DO 2115 IY=1, 200
            PROB(IX, IY)=PROB(IX,IY)*INVNMA
2115     CONTINUE
2120  CONTINUE
C  Finally, compute the entropy S
      S=0.0D0
      print*,'computing S'
      DO 2130 IX=1, 200
         DO 2125 IY=1, 200
            IF(P.ne.0d0)print*,'P=',P
            P=PROB(IX,IY)
            IF(P.EQ.0D0) GOTO 2125
            S=S-P*DLOG(P)
2125     CONTINUE
2130  CONTINUE
C Now return to the main program to write S for this value of A
      RETURN
      END
```

APPENDIX C. PROGRAM MCNIDER

```
************************************************************************
    PROGRAM MCNIDE
c
c  This program
c     1) computes Monte Carlo particle velocity and position as a function
c        of time step, t+^^t, using the McNider algorithm, Kamada vertical
c         velocity skewness variations, and Kamada mesoscale windflow and
c         turbulence simulation model.
c
c      2) computes the chaos metrics, Da (self-affine fractal dimension)
c         for velocity, S (information entropy), and lambda (the Lyapunov
c        exponent), as well as Ri, the atmospheric Richardson number, BVF
c         (Brunt-Vaisala frequency) l (buoyancy length scale), σu,v,w, tne
c         velocity variances, TKE, the turbulence kinetic energy, Db
c         (self-similar fractal dimension for position).
c
************************************************************************
c
      DIMENSION  nyval(45), k(45), ipts(100), bi(2,1800)
      DOUBLE PRECISION Linv, lglngth(45, 100), lgk(45),
     &          vert(0:3700), r(0:3700), r1(0:3700), r2(0:3700),
     &          el, p(1000), entropy(101),  up, dn, lyap, pu, pv, pw,
     &          sum(45, 100), DavsL(8,100), pk, ans(16), xy(90)
      EQUIVALENCE (vert, r )
      DATA k/1, 2, 3, 4, 5, 6, 8, 9, 10, 12, 15, 16, 18, 20, 24, 25,
     &     30, 36, 40, 45, 48, 50, 60, 72, 75, 80, 90, 100,
     &    120, 144, 150, 180, 200, 225, 240, 300, 360, 400,
     &    450, 600, 720, 900, 1200, 1800, 3600/
      OPEN (1, FILE='McNideis.dat', RECL=255, STATUS='new', ERR=700)
      OPEN (2, FILE='McNidell.dat', RECL=255, STATUS='new', ERR=800)
      OPEN (3, FILE='DavsL.dat', RECL=255, STATUS='new', ERR=900)
      OPEN (4, FILE='SvsL.dat', RECL=255, STATUS='new', ERR=1000)
      OPEN (7, FILE='lyap.dat', RECL=255, STATUS='new', ERR=1100)
      OPEN (8, FILE='bifurct.dat', RECL=255, STATUS='new', ERR=1200)
      OPEN (9, FILE='initial.dat', RECL=255, STATUS='old', ERR=1300)
c
c  **********
c  INITIALIZE
c  **********
      READ (9,30)
30    FORMAT (///)
      READ (9,*) iis, ill, iDavsL, isvsL, ilyap, ibifurct, noskew,
     & iMcNid, iKamada, dt, ivert, iphase, iveloc, windspd2, z0,
     & Start, m2, iw, ix, upfactor, dnfactor, dwfactor, zinitial,
     & icount, iposi
      WRITE(6,*)'iis, ill, iDavsL, isvsL, ilyap, ibifurct, noskew,'
      WRITE(6,*)'iMcNid, iKamada,dt,ivert,iphase,iveloc, windspd2,z0,'
      WRITE(6,*)'Start, m2, iw, ix,upfactor, dnfactor,dwfactor,zinitial'
      WRITE(6,*)'icount, iposi'
      WRITE(6,*) iis, ill, iDavsL, isvsL, ilyap, ibifurct, noskew,
     & iMcNid, iKamada, dt, ivert, iphase, iveloc, windspd2, z0,
     & Start, m2, iw, ix, upfactor, dnfactor, dwfactor, zinitial,
     & icount, iposi
c  Comments on input parameters:
c        iis      = 1 switch gives position and/or velocity file
```

```
c         iil      = 1 switch gives Ln ϵ vs Ln(L(ϵ) file
c         iDavsL   = 1 switch gives Da vs 1/L file
c         isvsL    = 1 switch gives entropy, S, vs 1/L file
c         ilyap    = 1 switch gives Lyapunov exponent vs 1/L file
c         ibifurct = 1 switch gives bifurcation disagram file
c         noskew   = 1 switch turns off vertical velocity skewness
c                      computation
c        iMcNid   = 1 switch turns on McNider vertical velocity skewness
c                       scheme
c         iKamada  = 1 switch turns on Kamada vertical velocity skewness
c                       scheme
c         dt       = time step size in seconds
c         ivert    = 1 switch gives vertical height file
c         iphase   = 1 switch gives phase space vector file
c         iveloc   = 1 switch gives velocity vector file
c         windspd2 = input proscribed wind speed at 2 meters
c         zi       = input proscribed fully turbulent boundary layer
c                      height
c        z0       = input proscribed surface roughness height (~1/7) the
c                    average height of surface roughness elements if
c                      element
c                   density is between 10 and 50% (like most vegetation)
c
c  Do some idiot proofing
c
      IF ( ivert .eq. 1) THEN
         iphase = 0
         iveloc = 0
         iposi  = 0
         ENDIF
      IF ( iphase .eq. 1) THEN
         ivert = 0
         iveloc = 0
         iposi  = 0
         ENDIF
      IF ( iposi .eq. 1) THEN
         ivert = 0
         iphase = 0
         iveloc = 0
         ENDIF
      IF ( noskew .eq. 1) THEN
         iMcNid = 0
         iKamada= 0
         ENDIF
      IF ( iMcNid .eq. 1) THEN
         noskew = 0
         iKamada= 0
         ENDIF
      IF ( iKamada.eq. 1) THEN
         noskew = 0
         iMcNid = 0
         ENDIF
c  Initialize variables
      m1 = 1
      n    = 3600
      en   = n
      WRITE (4,*)'     S         1/L'
      WRITE (7,*)'    Lyap       1/L'
      WRITE (8,*) '    1/L      bifurcation pt'
      DO 500 l = m1, m2
c  Let Linv, the inverse Obukhov length, 1/L = -kgw'Θ'/(Θ*ustar**3), be
```

```
c  varied in increments along the interval from -1 to 1.
         em2 = m2
         Linv = start + 0.3d0*(l-1)/em2
         zi = 2000 - 6000*(Linv + .2)
         r(0)  = 0.
         r1(0) = 0.
         r2(0) = 0.
         vert(0) = zinitial
         x = 0.001
         y = 0.002
         z = zinitial
         udp = 0.001
         vdp = 0.002
         wdp = 0.003
c  ********
c  GET iteration series for a given 1/L
c  ********
         IF ( iis .ne. 1 ) GOTO 150
         pu = 0.0
         pv = 0.001
c        pw = -.01*alog10(l)
         pw = -0.001
         time = 0.0
         rmax = 0.0
         r2max = 0.0
         rmin = 0.0
         r2min = 0.0
         sigU2sum = 0.
         sigV2sum = 0.
         sigW2sum = 0.
         zsum = 0.
         BVFsum = 0.
         BLSsum = 0.
         sumdw = 0.
         tkebar = 0.
         wbuoy = 0.
         sumw = 0.
         sumwbuoy = 0.
d        WRITE(6,*)'rmax2 =',rmax2
         rmax2 = 0.
         rmin2 = 0.
         iflag = 0
         DO 100 i = 1, n + 100
            CALL McNid (x, y, z, udp, vdp, wdp, dt, noskew, iMcNid,
     &      iKamada, Linv, windspd2, deltaz, zi, z0, wbuoy, sumw, Ri,
     &      sumwbuoy, BVF, BLS, sigmau2, sigmav2, sigmaw2, tke, icount,
     &      pu, pv, pw, i, time, wk, ustar, iw, ix, upfactor, iflag,
     &      sigmau, sigmav, sigmaw, Tlw, sumdw, dnfactor, dwfactor )
            BVFsum = BVFsum + BVF
            BLSsum = BLSsum + BLS
            sigU2sum  = sigU2sum + sigmau**2
            sigV2sum  = sigV2sum + sigmav**2
            sigW2sum  = sigW2sum + sigmaw**2
            zsum = zsum + z
            bltime = 0.3*zi/(sigmaw*3700.)
            eddytime = 0.01*Tlw
c           IF (Linv .ge. 0.000) dt = MIN (eddytime, bltime)
c           IF (Linv .ge. 0.000) dt = MIN (dt, 0.25)
c           IF (Linv .gt. 0.000) dt = eddytime
c           IF (Linv .lt. -0.00) dt = MIN (1.2*zi/(3700.0*wk), eddytime)
            dt = 0.166666667
            time = time + dt
cd          WRITE (6,*) 'time',time
c  get position vector, r1
            r1(i) = sqrt( x*x + y*y + (z-zinitial)**2 )
c  get velocity vector, r2
            r2(i) = sqrt( udp*dup + vdp*vdp + wdp*wdp )
c  get vertical position series
            IF ( ivert .eq. 1) r(i) = z - zinitial
c  get position series
            IF ( iposi .eq. 1) r(i) = r1(i)
c  get phase space position
            IF (iphase.eq. 1) r(i) = sqrt( r1(i)*r1(i) + r2(i)*r2(i) )
c  get velocity series
            IF (iveloc .eq. 1) r(i) = r2(i)
c  get min/max range of the values
            IF ( i .gt. 100 .and. r(i) .gt. rmax ) rmax = r(i)
            IF ( i .gt. 100 .and. r(i) .lt. rmin ) rmin = r(i)
            IF ( i .gt. 100 .and. r(i) .gt. rmax2) rmax2= r(i)
            IF ( i .gt. 100 .and. r(i) .lt. rmin2) rmin2= r(i)
c           WRITE (1,50) i,  r1(i), r2(i), r(i)
c50           FORMAT (1x, I4, 3(3x, f12.5) )
100         CONTINUE
150         CONTINUE
          WRITE (6,*)'iteration series created for 1/L = ', Linv
          IF ( ibifurct .ne. 1 ) GOTO 240
c  ***************
c  GET BIFURCATION diagram, using the last 1600 points.  For each 1/L
value
c  put all the unique points in "bi"
c  ***************
c
            m = 1
            bi(1,m) = Linv
            bi(2,m) = r(1999)
            slop = 0.01*(rmax - rmin)
c           IF (iveloc .eq. 1) slop = 0.005*11.
c           IF (ivert  .eq. 1) slop = 0.005*zi
c           IF (iphase .eq. 1) slop = 0.002*3000.
c     Check every r(i)
            DO 220 i = 2000, n+99
c     Check each r(i+1) against all unique points in bi
              DO 180 j = 1, m
                up = bi(2,j) + slop
                dn = bi(2,j) - slop
                IF ( r(i+1) .gt. dn .and. r(i+1) .lt. up ) GOTO 220
c     Since r(i+1) lies beyond margins of all "bi", we assume it a new
c     unique point, so add it to the list of values in "bi".
180             CONTINUE
            bi(2,m+1) = r(i+1)
            bi(1,m+1) = Linv
            m = m+1
220         CONTINUE
c         IF ( iveloc .ne. 1 ) bi(2,m) = r(1999)
          WRITE (8,230) (( bi(i,j), i = 1,2), j = 1,m)
230       FORMAT ( 1x, f9.5, 5x, f9.5 )
240       CONTINUE
          WRITE (6,*)'bifurcation diagram points found for 1/L = ', Linv
          IF ( ilyap .ne. 1 ) GOTO 260
c ************
c GET LYAPUNOV exponent for this series,
c ************
```

```
c                                 n
c      lyap = lim (1/n) sum log |f'(x    )|   , where f'(x    ) =μ(1 -2x  )
c             n>∞        i=1    e     i+1                  i+1          i+1
c
c    where we start from y(100) to avoid dependence on initial condition,
c     x0.
c
          lyap = 0.0
          DO 250 i = 1, n
c            lyap = lyap + dlog( abs(mu*(1.0 - 2.*y(i+100) ) ))
             lyap = 0.0000000000
250          CONTINUE
c    Normalize the lyapunov exponent by N - 100, the number of points
c    scanned.
          lyap = lyap/en
          WRITE (7,255) lyap, Linv
255       FORMAT (1x, 2(f9.5,2x) )
260          CONTINUE
          IF ( isvsL .ne. 1 ) GOTO 290
c
c ***********                          N
c GET ENTROPY value, S = - sum p log p   , for this iteration series & 1/L
c ***********                    i=1  i   e i
c
c  Zero out the number of points in each bin
          DO 265 ii = 1, 100
             ipts(ii) = 0
265          CONTINUE
c     Subdivide the interval into 100 sub-intervals,
c     ii, and check all pts(i), from 101 - 3600.
          entropy(1) = 0.0
          DO 270 i = 101, n + 100
             ii = 99*( r(i) - rmin2)/(rmax2 - rmin2) + 1.01
c     Make sure we don't create extra sub-intervals
             IF (ii .gt. 100) ii = 100
c     Increment the ipts counter for sub-interval ii for every r(i) found
c     in ii.
             ipts(ii) = ipts(ii) + 1
270          CONTINUE
          DO 285 ii = 1, 100
c     If no points are in sub-interval, ii, leave the entropy unchanged.
             IF ( ipts(ii) .eq. 0)  THEN
                si = 0.0
                GOTO 280
                ENDIF
c     Otherwise compute probability of a point being in ii as the # of
c     points in ii divided by N, the total number of points scanned.
             pts = ipts(ii)
             prob = pts/en
             si = prob*alog(prob)
c     Since the probability is less than or equal to 1, the log is
c     negative, so si is less than or equal to zero, so the entropy will
c     grow more positive, if the r(i) are scattered over more than one
c     sub-interval.
280          entropy(1) = entropy(1) - si
285          CONTINUE
          WRITE (4,255) entropy(1), Linv
          IF ( ill. eq. 1) WRITE (2, *)'   Ln(ε)   Ln [L(ε)]'
          WRITE (6,*)'entropy value found for 1/L = ', Linv
290       CONTINUE
          IF ( iDavsL .ne. 1 ) GOTO 500
```

```
c *******
c GET Da. First get total length as sum of y(i) changes for varying
c ******* spacings of i.
         nn = 45
         DO 400 j = 1, nn
            sum(j,l) = 1.0000
            DO 300 i = k(j), 3600, k(j)
               sum(j,l) = sum(j,l) + abs( r(i+100) - r(i-k(j)+100 ))
300            CONTINUE
c    Get log of the total length
            lglngth(j,l) = dlog10( sum(j,l) )
c    Get log of the spacing
            pk = k(j)
            lgk(j)  = dlog10( pk )
c    Put these log values into an array and send to the linear regression
c    routine.
            xy(2*j-1) = lgk(j)
            xy(2*j) = lglngth(j,l)
            IF (ill .eq. 1) WRITE (2,350) lgk(j), lglngth(j,l)
350         FORMAT (1x, f8.5, 3x, f8.5)
400         CONTINUE
c    WRITE (6,*) 'program passed statement 400'
c
c    DO LINEAR REGRESSION to get Da from log10(k) vs. log10 [L(ϵ)] values
c
c    Use only spacings up to 1200 points for the linear regression.  So
c    skip spacings 1800 and 3600.
         nn = nn - 2
         CALL STLNRG(nn, nyval, xy, ans)
c
c  -ans(2) contains Da
c  ans(14) contains the standard deviation of Da
c
c  Put 1/L, Da, σDa, BVF, BLS, σw², tke in a 7 by 1 array
c
         ptsinv = 1./(n+100.)
         BVFbar = BVFsum*ptsinv
         BLSbar = BLSsum*ptsinv
         sigw2bar = sigw2sum*ptsinv
         tkebar = .5*(sigu2sum + sigv2sum + sigw2sum)*ptsinv
         zbar = zsum*ptsinv
         DavsL(1,l) = -ans(2)
         DavsL(2,l) = -ans(14)
         DavsL(3,l) = Linv
         DavsL(4,l) = BVFbar
         DavsL(5,l) = BLSbar
         DavsL(6,l) = sigw2bar
         DavsL(7,l) = tkebar
         DavsL(8,l) = zbar
         WRITE (6,*)'Da value found for 1/L = ', Linv
500      CONTINUE
      WRITE (3,*)'    Da         σDa          Linv        BVF        BLS'
     &          ,'         σw2    tke   zavg'
      WRITE (6,*)'    Da         σDa          Linv        BVF        BLS'
     &          ,'         σw2    tke   zavg'
      WRITE (3,550) ((DavsL(i,l), i = 1,8), l = m1,m2)
      WRITE (6,550) ((DavsL(i,l), i = 1,8), l = m1,m2)
550   FORMAT (1x, 5(f10.4,1x), 2f7.2, f7.0)
      STOP
700   WRITE (6,*)'error opening McNideis.dat'
      STOP
```

```
800    WRITE (6,*)'error opening McNidell.dat'
       STOP
900    WRITE (6,*)'error opening DavsL.dat'
       STOP
1000   WRITE (6,*)'error opening SvsL.dat'
       STOP
1100   WRITE (6,*)'error opening Lyap.dat'
       STOP
1200   WRITE (6,*)'error opening bifurct.dat'
       STOP
1300   WRITE (6,*)'error opening initial.dat'
       STOP
       END
************************************************************************
       SUBROUTINE STLNRG (k, nyval, xy, ans)
c
c  This subroutine computes the slope and other statistics for a
c  linear regression with several y values for each x value or with
c  one independent variable.
c
c  argument   use     description
c     k       input   Number of different x values
c
c     nyval   input   Array of number of y values for each x
c                     value. Dimensioned k
c     xy      input   array of x and y pairs (x1, y1, x2, y2, etc.)
c                     of dimension (2,k)
c
c     ans     output  Array of results computed, dimensioned 16.
c
c                     ans
c                      1 Number
c                     *2 Slope
c                      3 Mean of y
c                      4 Mean of x
c                      5 y-intercept
c                      6 Sum of y**2
c                      7 Sum of squares mean
c                      8 Sum of squares slope
c                      9 Residual
c                     10 standard deviation of x
c                     11 standard deviation of y
c                     12 standard deviation error
c                     13 standard deviation y-bar
c                    *14 standard deviation slope
c                     15 standard deviation y-intercept
c                     16 F-Ratio for slope
c
************************************************************************
       DIMENSION nyval(1)
       DOUBLE PRECISION xy(1), ans(16)
c      WRITE (6,*)' program entered linear regression routine'
       sumx = 0.0
       sumy = 0.0
       sumx2 = 0.0
       ans(6) = 0.0
       sumxy = 0.0
       n = 0
       nx = 1
       ny = 2
       DO 20 j = 1,k
```

```
      nyval(j) = 1
      m = nyval(j)
      n = n + m
      DO 10 i = 1, m
      sumx = sumx + xy(nx)
      sumx2 = sumx2 + xy(nx)*xy(nx)
      sumy = sumy + xy(ny)
      ans(6) = ans(6) + xy(ny)*xy(ny)
      sumxy = sumxy + xy(nx)*xy(ny)
10    ny = ny + 1
      nx = nx + nyval(j) + 1
20    ny = ny + 1
      en = n
      ans(1) = en
      s1 = ans(1)*sumx2 - sumx*sumx
      s2 = ans(1)*sumxy - sumx*sumy
      en1 = en - 1.0
      en2 = en1 - 1.0
      ans(2) = s2/s1
      ans(3) = sumy/en
      ans(4) = sumx/en
      ans(5) = ans(3) - ans(2)*ans(4)
      ans(7) = ans(3)*sumy
      ans(8) = ans(2)*s2/en
      ans(9) = ans(6) - ans(7) - ans(8)
      s4 = ans(9)/(en - 2.0)
      endsl = en/s1
      ans(10) = sqrt(1.0/(en1*endsl))
      ans(11) = sqrt((ans(6) - ans(7))/en1)
      ans(12) = sqrt(s4)
      a13 = s4/en
      ans(13) = sqrt(a13)
      a14 = s4*endsl
      ans(14) = sqrt(a14)
      ans(15) = sqrt(a13 + a14*ans(4)*ans(4) )
      ans(16) = ans(8)/s4
      RETURN
      END
***********************************************************************
      SUBROUTINE McNid (x, y, z, udp, vdp, wdp, dt, noskew, iMcNid,
     & IKamada, Linv, windspd2, deltaz, zi, z0, wbuoy, sumw, Ri,
     & sumwbuoy, BVF, BLS, sigmau2, sigmav2, sigmaw2, tke, icount,
     & pu, pv, pw, i, time, wk, ustar, iw, ix, upfactor, iflag,
     & sigmaudp, sigmavdp, sigmawdp, Tlw, sumdw, dnfactor, dwfactor)
c  this subroutine computes particle velocity and position for different
c  values of Ri#, etc.
c
***********************************************************************
      REAL Km, L, lmdamu, lmdamv, lmdamw
      DOUBLE PRECISION Linv, ruu, rvv, rww, pu, pv, pw
c Get mesoscale and boundary layer flow parameters
c Initialize variables
      zi = 2000 - 6000*(Linv + .2)
      zovrzi = z/zi
      zovrL  = z*Linv
      ziovrL = zi*Linv
      abszovrL = abs(zovrL)
      L = 1./Linv
      CALL MESOFLOW( z, zi, z0, Linv, windspd2, ustar, Km, Ri,
     &     u, v, w, Theta2, dudz, dvdz, wk, D, R, wptpZ, BVF, iw,
     &     BLS, sigmaU2, sigmaV2, sigmaW2, tke, Wz, i, wk, ustar,
     &     bdthetdz,Thstar, Deltheta)
d     IF(i.lt.iw)WRITE(6,*)'zovrzi,zovrL,ziovrL',zovrzi,zovrL,ziovrL
c Get A, lambdas, σu,v,wdp, and buoyancy length scale, BLS
      IF ( zovrL .lt. 0.0 ) THEN
         sigmaudp = ustar*(12.0 + 0.5*zi/abs(L))**0.3333333
d        IF(i.lt.iw)WRITE (6,*)'σu''=u*(12+.5zi/|L|)**1/3',
d    &                  sigmaudp,ustar,zi,L
         IF ( abszovrL .le. 1. )
     &     A = 0.31*(1. - 3.*zovrL)**(-0.333333)*(1. - 15.*zovrL)**0.25*
     &         (0.55 + 0.38*zovrL )
         IF ( 0.1*abs(ziovrL) .gt. abszovrL .and. abszovrL .gt. 1.)
     &     A = 0.05*(1. - 3.*zovrL)**(-0.333333)*(1. - 15.*zovrL)**0.25
           A = MAX ( A, 0.07)
           A = MIN ( A, 0.18)
         lmdamu = 1.5*zi
         IF ( z .le. 0.1*zi .and. abs(zovrl) .gt. 1.) lmdamw = 5.9*z
         IF ( z .le. abs(L) ) lmdamw = z/( 0.55 + 0.38*zovrL )
         IF ( 0.1*zi .lt. z .and. z .lt. zi )
     &      lmdamw = 1.8*zi*( 1. - exp(-4.0*zovrzi)
     &                 - 0.0003*exp( 8.0*zovrzi ) )
         sigmawdp = Km/(A*lmdamw)
d        IF(i.lt.iw)
d    &      WRITE(6,*)'at z,σw''=Km/(A*lmdamw)',z,sigmawdp,Km,A,lmdamw
      ELSEIF ( zovrL .ge. 0.) THEN
         sigmaudp = 2.3*ustar
         lmdamu = 0.7*sqrt(zovrzi)*zi
         es = 70.0
         IF (z.lt.205.0) es = 0.35*z
         lmdamw = MAX( z, 2.9*es)
         Ric = 0.25
         Rifactor = ((Ric - 1.25*Ri)/Ric)**0.58
d        IF(i.lt.iw)WRITE(6,*)'Rif=((Ric-Ri)/Ric)**.58',Rifactor,Ric,Ri
         shear = sqrt(dudz**2 + dvdz**2)
         sigmawdp = 1.2*es*Rifactor*shear
d        IF(i.lt.iw)WRITE(6,*)'σw''=1.2esRi,shear',
d    &              sigmawdp, es, Rifactor, shear
         BLS = sigmawdp/BVF
         ENDIF
      sigmavdp = sigmaudp
      IF (ikamada .eq. 1) THEN
c  Use values derived from Sorbjan and Kamada from mesoscale subroutine
d       IF(i.lt.iw)WRITE(6,*)'σu²,σv²,σw²',sigmaU2,sigmaV2,sigmaW2
         sigmaudp = sqrt(sigmaU2)
         sigmavdp = sqrt(sigmaV2)
         sigmawdp = sqrt(sigmaW2)
d       IF(i.lt.iw)WRITE(6,*)'iKmda σu,v,w''',sigmaudp,sigmavdp,sigmawdp
c  To get vertical integral length, see pps. 244-247 Lectures on Air
c  Pollution
         aln = 0.41*z
         lmdamw = 0.27*sigmawdp/BVF
         lmdamw = 1./(1./aln + 1./lmdamw)
d       IF(i.lt.iw)WRITE(6,*)'iKmda aln,lmdamw',aln,lmdamw
      ENDIF
      IF (noskew .eq. 1) tke = 0.5*(sigmaU2 + sigmaV2 + sigmaW2)
      lmdamv = lmdamu
d     IF(i.lt.iw)WRITE(6,*)'lmdamu,lmdamv,lmdamw',lmdamu,lmdamv,lmdamw
c Get mean velocity
      V = sqrt( u*u + v*v + w*w )
c Get betas
d     IF(i.lt.iw) WRITE (6,*)'u,v,w,sigmaudp',u,v,w,sigmaudp
d     IF(i.lt.iw) WRITE (6,*)'sigmavdp,sigmawdp',sigmavdp,sigmawdp
```

```
      betau = 0.6*V/sigmaudp
      betav = 0.6*V/sigmavdp
      betaw = 0.6*V/sigmawdp
d     IF(i.lt.iw)WRITE(6,*)'at z,ßw=.6V/σw''',z,betaw,V,sigmawdp
c Get Lagrangian time scales
d     IF(i.lt.iw)WRITE(6,*)'betau,lmdamu,V',betau,lmdamu,V
      Tlu = 0.2*betau*lmdamu/V
      Tlv = 0.2*betav*lmdamv/V
      Tlw = 0.2*betaw*lmdamw/V
d     IF(i.lt.iw)WRITE(6,*)'Tlw=.2*ßw*lmdamw/V',z,Tlw,betaw,lmdamw,V
      IF ( ikamada .eq. 1) THEN
         IF ( Linv .ge. 0. ) Tlw = lmdamw/sigmawdp
         IF ( Linv .lt. 0. ) THEN
            Tlw = 0.3*zi/wk
            AspectR = 2.0 - 40.0*Linv
            Tlu = AspectR*Tlw
            Tlv = Tlu
            ENDIF
d     IF(i.lt.iw)WRITE(6,*)'Tlw=lmdamw/σw''',Tlw,lmdamw,sigmawdp
         ENDIF
      IF ( Tlu .lt. 0.02*dt ) Tlu = 0.02*dt
      IF ( Tlv .lt. 0.02*dt ) Tlv = 0.02*dt
      IF ( Tlw .lt. 0.02*dt ) Tlw = 0.02*dt
c Get Autocorrelation function
d     IF(i.lt.iw)WRITE(6,*)'dt,TLu,TLv,TLw', dt, Tlu,Tlv,TLw
      Ru = exp( -dt/Tlu)
      Rv = exp( -dt/Tlv)
      Rw = exp( -dt/Tlw)
d     IF(i.lt.iw)WRITE (6,*)'at z,Rw=exp(-dt/Tlw)',z,Rw,dt,Tlw
d     IF(i.lt.iw)WRITE (6,*)'Ru,Rv,Rw,ikamada',Ru,Rv,Rw,ikamada
c Get standard deviations of velocity
      sigmautp = sigmaudp*sqrt( (1. - Ru**2) )
      sigmavtp = sigmavdp*sqrt( (1. - Rv**2) )
      sigmawtp = sigmawdp*sqrt( (1. - Rw**2) )
d     IF(i.lt.iw)
d    &   WRITE(6,*)'at z,σw'''=σw''√(1-Rw²)',z,sigmawtp,sigmawdp,Rw
d     IF(i.lt.iw)
d    &WRITE(6,*)'σu''',σv''',σw''',sigmautp,sigmavtp,sigmawtp
c Get random normal fluctuation velocities, utp & vtp, at time, t-dt
      ruu = abs(pu/3.d0)
      rvv = abs(pv/3.d0)
      rww = abs(pw/3.d0)
d     IF(i.lt.iw) WRITE (6,*)'ruu,rvv,rww',ruu,rvv,rww
      call NORNG( ruu, pu )
      call NORNG( rvv, pv )
      IF( pw .ge. 0.0 )wtp = upfactor*(pw*(sigmaw2 - sigmawtp) + wbuoy)
      utp = sigmautp*pu
      vtp = sigmavtp*pv
      IF ( noskew .ne. 1 .and. Linv .lt. 0. ) GOTO 250
         call NORNG( rww, pw )
         wtp = sigmawtp*pw + wbuoy
d     IF(i.lt.iw) WRITE (6,*)'pu,pv,pw',pu,pv,pw
d     IF(i.lt.iw) WRITE (6,*)'utp,vtp,wtp',utp,vtp,wtp
d     IF(i.lt.iw) WRITE (6,*)'w''' = σw'''*pw',wtp,sigmawtp,pw
         GOTO 500
250   CONTINUE
d     IF(i.lt.iw) WRITE (6,*)'pu,pv,pw',pu,pv,pw
d     IF(i.lt.iw) WRITE (6,*)'utp,vtp,wtp',utp,vtp,wtp
      if ( iMcNid .ne. 1 ) GOTO 400
c modify wtp for skewness according to McNider method
      iwp = 0
      iwm = 0
300   CONTINUE
c Get random normal fluctuation velocity, wtp, at time, t-dt
         call NORNG( rww, pw )
         wtp = sigmawtp*pw
         IF ( wtp .ge. 0. .and. iwp .eq. 0 ) wp = wtp
         IF ( wtp .ge. 0. ) iwp = 1
         IF ( wtp .le. 0. .and. iwm .eq. 0 ) wm = wtp
         IF ( wtp .le. 0. ) iwm = 1
         IF ( iwp .eq. 0 .or. iwm .eq. 0 ) GOTO 300
c Get S function
      IF ( zovrL .gt. 0. )S = 0.1 -  0.2*zovrL**0.2
      IF ( zovrL .le. 0. )S = 0.1 + (0.6/( 0.68*(1.-15.*zovrL)**(-0.25)
     &     - 1.8*zovrL ) )
d     IF(i.lt.iw) WRITE (6,*)'iwp,iwm,S',iwp,iwm,S
c Get alpha and eta
      alpha = - 0.028 - 0.6*abs(S)
      eta   =   0.54*abs(S)
c Get wtp(t-dt) !!! Pielke prints this on p. 178 as wdp(t-dt) CHECK THIS
c by looking at McNider paper!!!!!
      IF ( zovrL .le. 0. ) wtp = alpha*wp + wm/alpha - eta
      IF ( zovrl .gt. 0. ) wtp = wp/alpha + alpha*wm + eta
d     IF(i.lt.iw) WRITE (6,*)'alpha,eta,wtp',alpha,eta,wtp
      GOTO 500
400   CONTINUE
      IF ( iKamada .ne. 1 .and. Linv .lt. 0. ) GOTO 500
c
c Try Kamada-Berkowicz double gaussian skewness approx instead of McNider.
c This approach assumes that mean absolute vertical fluctuation velocity
c is ~ 0.55wk, vertically averaged over the BL depth, where wk generalizes
c w* to mixed forced/free convection.  We let the up/downdraft volume
c ratio in the convective BL ~ 0.4/0.6, with mean updraft velocity
c ~ 1.08σw and the mean downdraft velocity ~ 0.87σw.  We assume that the
c two gaussian velocity distributions are displaced from zero by this
c amount with 60% of the distribution in the downdraft volume and 40%
c in the updraft volume.  We assume that departures from Gaussian are
c small for stable cases. (See pps. 199-201, Ch. 4 Lectures on Air
c Pollution Modeling)
c
         utp = pu*sigmaudp*sqrt(1 - Ru**2)
         vtp = pv*sigmavdp*sqrt(1 - Rv**2)
         wbuoy = Rw*wbuoy - (9.801/Theta2)*Deltheta*dt
         sumwbuoy = sumwbuoy + wbuoy
      call NORNG( rww, pw )
      IF ( pw .ge. 0.0 ) ikount = ikount + 1
d        IF(i.lt.iw) WRITE (6,*)'ikount,icount',ikount,icount
      IF ( ikount .eq. icount ) THEN
         pw = -pw
         ikount = 0
         ENDIF
c The above sets an (icount + 1)/(2*icount) chance of updraft & an
c                   (icount - 1)/(2*icount) chance of a downdraft.
c E.g., if icount = 5, then updraft/downdraft chances are 60%/40%.
420   CONTINUE
c     IF( pw .ge. 0.0 )wtp = upfactor*sigmawtp*pw + wbuoy
c     IF( pw .le. 0.0 )wtp = dnfactor*sigmawtp*pw + wbuoy
      IF( pw .ge. 0.0 )
     &      wtp = upfactor*pw*sigmawdp*sqrt(1 - Rw**2) + wbuoy
      IF( pw .le. 0.0 )
     &      wtp = dnfactor*pw*sigmawdp*sqrt(1 - Rw**2) + wbuoy
c     IF( pw .le. 0.0 ) wtp = upfactor*pw*sigmawdp + wbuoy
```

```
c      IF( pw .le. 0.0 ) wtp = dnfactor*pw*sigmawdp + wbuoy
       sumw = sumw + wtp
d      IF(i.lt.iw)WRITE(6,*)'wtp,pw',wtp,pw
d      IF(iflag .eq. iw)
d     &     WRITE(6,*)'wbuoy,Rw,δΘ,sumw', wbuoy,Rw,Deltheta,sumw
d      IF(iflag .eq. ix) WRITE(6,*)'sumwbuoy,sumw', sumwbuoy,sumw
500    CONTINUE
c
c Get fluctuation velocities
       udp = udpold*Ru + utp
       vdp = vdpold*Rv + vtp
       wdp = wdpold*Rw + wtp
       IF (ikamada .eq. 1) THEN
          udp = udpold + utp - (1. - Ru)*udpold
          vdp = vdpold + vtp - (1. - Rv)*vdpold
          wdp = wdpold + wtp - (1. - Rw)*wdpold
       ENDIF
d      IF(i.lt.iw)
d     &    WRITE (6,*)'at z,wdp=wdpold*Rw+wtp',z,wdp,wdpold,Rw,wtp
c Get position
       x = x + udp*dt
       y = y + vdp*dt
       z = z + wdp*dt
       IF ( z .lt. 10*z0 ) THEN
          z = abs(z) + 10.*z0
          wdp = abs(wdp)
       ELSEIF ( z .gt. zi ) THEN
          z = 2.0*zi - z
          z = MAX(z, 0.9*zi)
          wdp = -abs(wdp)
       ENDIF
       udpold = udp
       vdpold = vdp
       wdpold = wdp
       iflag = iflag + 1
       IF (iflag .gt. ix) THEN
d         WRITE (6,1110)time,Linv,Ri,udp,vdp,wdp,x,y,z
d         WRITE (6,1120)Km, V, ustar, wpTpZ, sigmaW2, tke, Tlw, Rw, Wz
          iflag = 1
          ENDIF
************************************************************************
c  debug formats
d1000 FORMAT (10x, 2g11.3)
d1010 FORMAT (10x, 3f11.3)
d1020 FORMAT (10x, 4f11.3)
d1030 FORMAT (1x, i5 )
d1040 FORMAT (10x, 3g11.3, i5)
d1050 FORMAT (10x, 4f11.3, i5)
d1055 FORMAT (10x, 5f11.3, i5)
d1060  CONTINUE
d1110 FORMAT (1x,'tm 1/L Ri u v w x y z ',f6.1,2f6.2,3f6.1,3f7.1)
d1120 FORMAT (1x,'Km V u* Q σw² e Tw Rw Wz ',f6.1,5f6.2,f6.1,2f6.2)
       RETURN
       END
************************************************************************
       SUBROUTINE MESOFLOW ( z, zi, z0, Linv, windspd2, ustar, Km, Ri,
     &      u, v, w, Theta2, dUtotdz, dvdz, wk, D, R, wpTpZ, BVF, iw,
     &      BLS, sigmaU2, sigmaV2, sigmaW2, tke, Wz,i,wk, ustar,
     &      bdthetdz, Thstar, Deltheta)
c This subroutine computes the mesoscale flow parameters used as
c inputs to the standard mcnider particle calculations.  It also computes
```

```
c some alternate values for the σu, v, w, and other quantities.
      REAL k, Ll, lnzovz0, LovZi, Km, L, lepsilon, lk
      DOUBLE PRECISION Linv
      DATA g, k, z1, f / 9.801, 0.4, 2.0, r.0001 /
d     IF(i.lt.iw)WRITE(6,*)'in MESOFLOW i = ',i
c  Zero out some variables
      u = 0.
      v = 0.
      w = 0.
      dUdZ = 0.
      dVdZ = 0.
      wstar = 0.0
      Wk = 0.
      Ri = 0.
      Km = 0.
      tke = 0.
      BVF = 0.
      BLS = 0.
c  Initialize some values
      D = 0.1
      R = 0.2
      z0inv = 1./z0
      lnzovz0 = alog(z*z0inv)
      IF ( z .lt. z0 ) z = z + z0
      zinv = 1./z
      ziinv = 1./zi
      L = 1./Linv
      gzi = 9.801*zi
      z1ovL = z1*Linv
      Theta2 = 290.0
      zovL = z*Linv
c  Get psim & psih for surface layer.  If stable,
      psim = -5.*z1ovL
      psih = psim
d     IF(i.lt.iw)WRITE(6,*)'psim=psih=-5z1/L',psim,psih,z1ovL
c  If unstable,
      IF ( Linv .lt. 0.) THEN
           phim = (1. - 28.*z1ovL)**(-0.25)
d     IF(i.lt.iw)WRITE(6,*)'phim=(1-28z1/L)**(-¼)',phim,z1ovL
c        Get psim by Kamada (unpub)
           IF ( z1ovL .lt. 0 .and. z1ovL .gt. -0.01) z1ovL = -0.01
d     IF(i.lt.iw)WRITE(6,*)'just before psim = '
           psim = (.0159651383 - 5.4151107*z1ovL)/
     &            (1. - 3.59002231*z1ovL -0.799168457*z1ovL**2)
d     IF(i.lt.iw)WRITE(6,*)'psim=(a-cz1/L)/(1-bz1/L-d(z1/L)²',psim,z1ovL
c        Also from Dyer and Bradley (1982) BLM 22, 3-19, we have for psi
H
d     IF(i.lt.iw)WRITE(6,*)'just before psih = '
           psih = 2.*alog( 0.5*(1. + sqrt( 1. - 14.*z1ovL )) )
d     IF(i.lt.iw)WRITE(6,*)'psih=2Ln(½(1+√(1-14z1/L)))',psih,z1ovL
      ENDIF
c  Get windspeed drag coefficient and friction velocity
d     IF(i.lt.iw)WRITE(6,*)'just before sqrtCdn = '
      sqrtCdn = 0.4/( alog(z1*z0inv) - psim)
d     IF(i.lt.iw)WRITE(6,*)'√Cdn=.4/(Ln(z1/z0)-psim)',sqrtCdn,z1,z0,psim
      ustar = MAX( sqrtCdn*windspd2, 0.01 )
d     IF(i.lt.iw)WRITE(6,*)'u*=MAX(√Cdn*V2,0.01)',ustar,sqrtCdn,windspd2
c
c  For dispersion, get vertical profiles of relevant variables, using
c  similarity theory from (Sorbjan ,Structure of the Atmos BL, Ch. 4) and
c  (Panofsky and Dutton, Atmos Turbulence).  First get oft used variables
```

```
      zovzi = z*ziinv
      z0ovL = z0*Linv
      ziovL = zi*Linv
      Lovzi = L*ziinv
      ustar2 = ustar*ustar
      ustar3 = ustar*ustar2
      wpTp0 = -0.25508*Linv*ustar3*Theta2
d     IF(i.lt.iw)WRITE (6,*)'wΘ= =-.255u**3Θ/L',wpTp0,ustar,Theta2,L
      Tstar = -wpTp0/ustar
      IF ( Linv .lt. 0.) THEN
         Wstar = (gzi*wpTp0/Theta2)**.33333
d        IF(i.lt.iw)WRITE(6,*)'w*=(gziwΘ/Θ)**1/3',wstar,gzi,wpTp0,Theta2
c           Replace w* with wk for mixed forced/free convection (Kamada,
c          NPS-PH-92-007.  Here, add only surface layer shear production.
         Ar = (1. - 150.0*z0ovL)**0.333333 - 1
d        IF(i.lt.iw)WRITE (6,*)'Ar= (1 - 150z0/L)**(1/3) - 1'
d        IF(i.lt.iw)WRITE (6,*)Ar, z0ovL
d     IF(i.lt.iw)WRITE(6,*)'just before Wk = '
         Wk = wstar*(1. - ( 3.125 - 2.5*alog(Ar) )*Lovzi )**0.33333
d        IF(i.lt.iw)WRITE (6,*)'Wk=w*(1-3.124-2.5ln(Ar))L/zi)**(1/3)'
d        IF(i.lt.iw)WRITE (6,*)wk,wstar,Ar,Lovzi
         Thstar = -wpTp0/wk
c  Decide what algos to use, based on z/L.
         ENDIF
      IF ( zovL .lt. -0.5 ) icase = 1
      IF ( zovL .lt. 0.0 .and. zovL .ge. -0.5 ) icase = 2
      IF ( zovL .ge. 0.01 ) icase = 3
      GOTO ( 330, 400, 430 ) icase
330      CONTINUE
*********
c Get Km, & σu,v,w for forced/free convective BL.  Ri and shear not
needed.
*********
c      Get potential temperature at height z above -L/2
         R = abs(R)
         zovzi13 = zovzi**.33333
         zovzi23 = zovzi13*zovzi13
         tp1 = 1.0 - zovzi
         tp4 = tp1 + D
         tp413 = tp4**.33333
         tp423 = tp4**.66667
         Thstar = -wpTp0/wk
         tp23 = tp1**.666667
         R23 = R**.666667
         Oldtheta = Theta
         Theta = Theta2 + Thstar*( tp23/zovzi13 + R23*zovzi23/tp413 )
         IF ( i.eq. 1) Oldtheta = Theta
         Deltheta = Theta - Oldtheta
d        IF(i.lt.iw)WRITE (6,*)'δΘ,Θ,OldΘ', Deltheta, Theta, Oldtheta
c     temperature and buoyancy fluxes at height, z
         wpTpZ = wpTp0*(1. - 1.2*zovzi)
         B = g/Theta2
         BwpTpZ = B*wpTpZ
c      Get vertical velocity variance at height z above -L/2
         sigmaW2 = wk**2*(1.1*zovzi23*tp23 + R23*zovzi23*tp423)
d     IF(i.lt.iw)WRITE (6,*)'σw²=wk²(1.1(z/zi)**2/3(1-z/zi)**2/3 + '
d     IF(i.lt.iw)WRITE (6,*)'R**(2/3)(z/zi)**(2/3)(1-z/zi+D)**(1/3)'
d     IF(i.lt.iw)WRITE (6,*)sigmaW2,wk,zovzi,R,D
         sigmaW = sqrt(sigmaW2)
c      Get TKE dissipation rate at height z above -L/2, using similarity
         tmp10 = Wk**3*ziinv
```

```
            epsilon0= 0.75*tmp10
            phiEp = 0.75*tp1 + R*zovzi
            epsilon = phiEp*tmp10
d       IF(i.lt.iw)WRITE (6,*)'ε = Φε*wk**3/zi',epsilon,phiEp,wk,zi
c        Get σu²,σv²
            wpsqovE = 0.333*( (2.*epsilon0 - epsilon)/epsilon0 +
      &                BwpTpZ/epsilon )
d       IF(i.lt.iw)WRITE(6,*)'w'²/e=(2ε0-ε)/3ε0 +ßw'Θ'z/ε',wpsqovE
d       IF(i.lt.iw)WRITE(6,*)'ε0, ßw'Θ'z', epsilon0, BwpTpZ
            sigmaU2 = 0.5555*sigmaW2*( 2./wpsqovE - 1.)
c           sigmaU2 = MAX( sigmaU2, .33333 )
            sigmaV2 = 0.8*sigmaU2
d       IF(i.lt.iw)WRITE (6,*)sigmaU2,wk,zovzi,sigmW2
c        The following tke parameterization is OK for low shear
            tke = 0.5*( sigmaU2 + sigmaV2 + sigmaW2 )
d       IF(i.lt.iw)WRITE (6,*)'e = σw²/2 + 3σu²/2',tke,sigmaW2,sigmaU2
            tke12inv = 1./sqrt(tke)
            tke32inv = tke12inv**3
c       Get dissipation & mixing length scales & vertical diffusion
c        coefficient
            lepsilon = 1./( zinv + 1.4*epsilon*tke32inv )
d      IF(i.lt.iw)WRITE(6,*)'lε = 1/(1/z +1.4ε√e )',lepsilon,z,epsilon,tke
            lk = MIN ( z, 3.0*lepsilon*sigmaW2/tke )
d       IF(i.lt.iw)WRITE (6,*)'lk = MIN(3lε*σw²/tke, z)',lk,lepsilon,tke,z
c       The mixing length scale and diffusion coefficients are Kamada
c       variants
            Km = 0.44*lk/tke12inv
d       IF(i.lt.iw)WRITE (6,*)'Km = 0.44*lk*sqrt(tke)',Km,lk,tke
c Get temperature and buoyancy gradients
c       dthetadz = .6*Thstar*ziinv*(tp23/zovzi23**2 -2.*R23*zovzi23/tp423)
        dthetadz = .6*Thstar*ziinv*(tp23/zovzi13    -2.*R23*zovzi23/tp423)
        Bdthetdz = B*dthetadz
        GOTO 460
400     CONTINUE
**********
c   get Km for unstable surface layer.  Shear and Ri not needed for
unstable.
**********
        Km = k*ustar*z/phim
        tkeus1 = (5.6*Km*zinv)**2
        Ri = zovL
        phi1 = ( 12. - 0.5*ziovL )**.333333
        phi2 = phi1
        phi3 = ( 1. - 14.*zovL )**.25
        phiEp= (1. + .75*(-zovL)**.6667)**1.5
        epsilon0 = ustar3*phiEp*zinv
        GOTO 440
430     CONTINUE
**********
c  get Km, Ri, and other parameters for neutral to stable BL
**********
            h = zi
            IF ( zi .lt. -98. ) h = sqrt(k*ustar*Linv/f)
d           IF(i.lt.iw)WRITE(6,*)'h=zi,or √(ku*/(fL))',h,zi,f
            hinv = 1./h
            zovH = z*hinv
            zsqovHL = zovL*zovH
            zsqovH2 = zovH*zovH
            alpha = 2.0 - 10.0*Linv
            beta  = 3.0 - 20.0*Linv
d           IF(i.lt.iw)WRITE(6,*)'α,ß,z/H',alpha, beta, zovH
```

```
          tp8 = 1. - zovH
          tp8alpha = tp8**alpha
d         IF(i.lt.iw)WRITE(6,*)'(1-z/H),(1-z/H)**α', tp8,tp8alpha
          tp8al2  = sqrt(tp8alpha)
          tp8beta = tp8**beta
d         IF(i.lt.iw)WRITE(6,*)'(1-z/H)**α/2,(1-z/H)**ß',tp8al2,tp8beta
          phi1 = 2.2*tp8al2
          phi2 = phi1
          phi3 = 1.6*tp8al2
c       local friction velocity
          ul = ustar*sqrt(tp8alpha)
d         IF(i.lt.iw)WRITE(6,*)'ul=u*(1-z/H)**(α/2)',ul,ustar,zovH
c       local Obukhov length
          Ll = L*tp8**(1.5*alpha - beta)
d         IF(i.lt.iw)WRITE(6,*)'Ll=L(1-z/H)**(3α/2-ß)',Ll,L,zovH,alpha,
d    &             beta
c       total shear
c         dUtotdz = 4.7*ul/(k*Ll)
          dUtotdz = 2.5*ustar*zinv*(1.+4.7*zovL)**(0.5*alpha)*tp8al2
d         IF(i.lt.iw)WRITE(6,*)'dU/dz=4.7ul/(kLl)',dUtotdz,ul,k,Ll
c      temperature and buoyancy fluxes at height, z
          wpTpZ = wpTp0*(1. - zovzi)**beta
c       Brunt Vaisala frequency at z
          tp9 = (1. + 3.7*zovL)*zinv
d         IF(i.lt.iw)WRITE(6,*)'tp9=(1+3.7z/L)/z',tp9,zovL,zinv
          BVF = 4.3*ustar*Tstar**2*tp9*tp8beta*tp8alpha
d         IF(i.lt.iw)WRITE(6,*)'BVF=4.3u*Θ*²(1+3.7z/L)(1-z/H)**(α+ß)/z',
d    &   BVF,ustar,Tstar,zovH,zovL,z
c       TKE dissipation rate at z
          phiEp = 3.6*tp9*tp8beta*zinv
          epsilon = phiEp*ustar3
c       Richardson number at z
          tp10 = 1./(1. + 5.*zovL )
          Ri = zovL*tp10
c       momentum/heat diffusivities at z
          Km = k*ustar*z*tp8*tp10
d         IF(i.lt.iw)WRITE(6,*)'Km=ku*z(1-z/H)/(1+5z/L)',Km,k,ustar,z,
d    &             zovH,zovL
*****************************
440       CONTINUE
c Get σu,v,w, tke, total horizontal velocity, & temperature
          sigmaU = ustar*phi1
          sigmaU2= sigmaU*sigmaU
          sigmaV = 0.894*sigmaU
          sigmaV2= 0.5*sigmaU2
          sigmaW = ustar*phi3
          sigmaW2 = sigmaW**2
          tke = 0.5*(sigmaU2 + sigmaV2 + sigmaW2)
d         IF(i.lt.iw)WRITE(6,*)'tke=σui²/2',tke,sigmaU2,sigmaV2,sigmaW2
c         IF ( icase .eq. 2 ) tke = tkeusl
          Oldtheta = Theta
          Theta =  Theta2 + 2.5*Tstar*( lnzovz0 - psih )
          IF ( i. eq. 1) Oldtheta = Theta
          Deltheta = Theta - Oldtheta
d         IF(i.lt.iw)WRITE (6,*)'δΘ,Θ,OldΘ', Deltheta, Theta, Oldtheta
460       CONTINUE
          Vtotal = 2.5*ustar*(lnzovz0 - psim)
d         IF(i.lt.iw)WRITE(6,*)'V=2.5u*(Ln(z/z0)-psim))',Vtotal,ustar,
d    &             lnzovz0,psim
          u = Vtotal
d         IF(i.lt.iw)WRITE(6,*)'Θ=Θ2+2.5Θ*(Ln(z/z0)-psih)',theta,theta2,
d    &               tstar,lnzovz0,psih
**********************************************************************
c  debug formats
d1000 FORMAT (10x, 2g11.3)
d1010 FORMAT (10x, 3f11.3)
d1020 FORMAT (10x, 4f11.3)
d1030 FORMAT (1x, i5 )
d1040 FORMAT (10x, 3g11.3, i5)
d1050 FORMAT (10x, 4f11.3, i5)
d1055 FORMAT (10x, 5f11.3, i5)
d1060  CONTINUE
      RETURN
      END
**********************************************************************
      SUBROUTINE NORNG(r,p)
c
c  This subroutine generates a sequence of numbers normally and randomly
c  distributed over the interval -3 to 3 from uniformly distributed random
c  numbers, by the method of linear approximation to the inverse of the
c  accumulative normal distribution function
c
      DOUBLE PRECISION r,p, y(6), x(6), s(5)
      DATA y/0.d0, 0.0228d0, 0.0668d0, 0.1357d0, 0.2743d0, 0.5d0 /,
     &     x/ -3.01d0, -2.0d0, -1.5d0, -1.0d0, -0.6d0, 0.0d0 /,
     &     s/ 43.8596d0, 11.3636d0, 7.25689d0, 2.891352d0, 2.65887d0 /
      CALL STRNUM(r)
      p = r
      i = 1
      IF ( p .gt. 0.5d0 ) p = 1.0d0 - r2     IF ( p .lt. y(i+1) ) GOTO 8
      i = i + 1
      GOTO 2
8     p = ( ( p - y(i) )*s(i) + x(i) )
      IF ( r .ge. 0.5d0 ) p = -p
cd    WRITE (6,*)'random numbers, r & p',r,p
      RETURN
      END
************************************************************************
    SUBROUTINE STRNUM(r)
c
c  This subroutine generates a sequence of numbers which are randomly and
c  uniformly distributed over the unit interval.
c
      DOUBLE PRECISION p1, r
      bb = 1.d0
      p1 = r*317.d0
c     WRITE (6,*)'p1',p1
      r  = abs( p1 - ( INT(p1/bb)*bb) )
      RETURN
      END
************************************************************************
      FUNCTION MAX (a, b)
      max = b
      IF ( a .gt. b) max = a
      END
************************************************************************
    FUNCTION MIN (a, b)
      min = b
      IF ( a .lt. b) max = a
      END
************************************************************************
```

LIST OF REFERENCES

Agee, A., T. Chen, and K. Dowell, 1973: A review of mesoscale cellular convection,. *Bul. Amer. Met. Soc.,* **54**, No. 10, pp. 1004-1012.

Andre, J.C., G. De Moor, P. LaCarrere, G. Therry, and R. du Vachat, 1978: Modeling the 24-hour evolution of the mean and turbulent structures of the planetary boundary layer.*, J. Atmos. Sci.* **35**, 1861-1883.

Baker, G., and J. Gollub, 1990: *Chaotic dynamics*, Cambridge University Press, Cambridge, MA.

Businger, J. A., 1973: Turbulent transfer in the atmospheric surface layer., *Workshop on micrometeorology,* ed. D. A. Haugen, American Meteorological Society, Boston, MA, pp. 67-100.

Collins, R., and P. Rastogi, 1989: Fractal analysis of gravity wave spectra in the middle atmosphere., *J. Atm. and Terr. Phys.,* **51**, No 11/12, pp. 997-1002.

DeCaria, A., 1992: *Multi-fractal analysis of nocturnal layer time series from the Boulder atmospheric observatory*, Master's Thesis, Naval Postgraduate School, Monterey, CA.

Devaney, R.L., 1990: *Chaos, fractals, and dynamics: computer experiments in mathematics,* Addison-Wesley Publishing Co., Menlo Park, CA.

Gleich, J., 1987: *Chaos: making a new science*, Viking Penguin, Inc., New York.

Gould, H., and J. Tobochnik, 1988: *An introduction to computer simulation methods, applications to physical systems, part 1*, Addison-Wesley Publishing Co., Reading, MA, pp. 151-178.

Grassberger, P., 1986: Estimating the fractal dimensions and entropies of strange attractors., *Chaos*, ed. A. Holden, Princeton University Press, Princeton, N.J., pp. 291-311.

Kamada, R.,*Amending the w. velocity scale for surface layer, entrainment zone, and baroclinic shear in mixed forced/free turbulent convection.*, NPS-PH-92-007, Naval Postgraduate School, Monterey, CA.

-----, 1992: unpublished notes, NPS particle dispersion model, Naval Postgraduate School, Monterey, CA.

-----, and A. DeCaria, 1992: *A self-affine multi-fractal wave/turbulence discrimination method using data from single point fast response sensors in a nocturnal atmospheric boundary layer.*, NPS-PH-92-008, Naval Postgraduate School, Monterey, CA.

Lenschow, D.H., J.C. Wyngaard, and W.T. Pennell, 1980: Mean-field and second-moment budgets in a baroclinic, convective boundary layer., *J. Atmos. Sci.*, **37**, 1313-1326.

Mason, P.J. and D.J. Thompson, 1987: Large eddy simulations of the neutral-static-stability planetary boundary layer., *Quart. J. Roy. Meteor. Soc.*, **113**, 413-443.

McHardy, I., and B. Czerny, 1987: Fractal X-ray time variability and spectral invariance of the Seyfert galaxy NGC5506., *Nature,* **325**, pp. 696-698.

Panofsky, H., and J. Dutton, 1984: *Atmospheric Turbulence*, Wiley, New York.

Pielke, R.A., 1984: *Mesoscale meteorological modeling,* Academic Press, Orlando, FL, pp. 173-186.

Sorbjan, Z., 1990: *Structure of the atmospheric boundary layer,* Prentice-Hall, Inc., Englewood Cliffs, N.J., pp. 67-124.

Stull, R., 1988: *An introduction to boundary layer meteorology*, Kluwer Academic Publishers, Dordrecht, The Netherlands.

Therry, G. and P. Lacarrere, 1983: Improving the eddy kinetic energy model for planetary boundary layer description., *Boundary Layer Meteorol.*, **25**, 63-88.

Tribus, M., and E. McIrvine, 1971: Energy and information., *Sci. Am.*, **224**, No. 3, pp. 179-188.

Venkatram, A., and J. Wyngaard, 1988: *Lectures on air pollution modeling*, American Meteorological Society, Boston, MA, pp. 39-61.

Wolf, A., 1986: Quantifying chaos with Lyapunov exponents., *Chaos*, ed. A. Holden, Princeton University Press, Princeton, N.J., pp. 273-290.

INITIAL DISTRIBUTION LIST

1.	Defense Technical Information Center Cameron Station Alexandria, VA 22304-6145	2
2.	Library, Code 52 Naval Postgraduate School Monterey, CA 93943-5000	2
3.	Chairman (Code PH/Wh) Department of Physics Naval Postgraduate School Monterey, CA 93943-5000	2
4.	Ray Kamada (Code PH/Kd) Department of Physics Naval Postgraduate School Monterey, CA 93943-5000	5
5.	MAJ Korey V. Jackson 1719 Seneca Leavenworth, KS 66048	2
6.	Ken Davidson (Code MR/Ds) Department of Meteorology Naval Postgraduate School Monterey, CA 93943-5000	1
7.	Mr. Bart Lundblad (Aerospace/El Segundo) PO Box 92957 Los Angeles, CA 90009	1